Numerical Methods for Non-local Operators

Steffen Börm

1:17, February 4, 2021

Contents

1 Introduction

Non-local operators appear naturally in a wide range of applications, e.g., in the investigation of gravitational, electromagnetic or acoustic fields. Handling non-local interactions poses an interesting algorithmic challenge: we consider classical gravitational fields as an example. In a first step, we consider two stars located at positions $x, \hat{x} \in \mathbb{R}^3$ with masses $m, \hat{m} \in \mathbb{R}_{>0}$. The gravitational force exerted by the star at position x on the star at position $\hat{x}$ is given by

$$\hat{f} = \gamma \hat{m} m \frac{x - \hat{x}}{\|x - \hat{x}\|_2^3}$$

where $\gamma \in \mathbb{R}_{>0}$ is the universal gravitational constant. If we consider only two interacting stars, we can evaluate this expression directly and efficiently and use it as the basis of simulations.

The situation changes significantly if we consider a larger number of stars: Let $n \in \mathbb{N}$, and let $x_1, \ldots, x_n \in \mathbb{R}^3$ be positions and $m_1, \ldots, m_n \in \mathbb{R}_{>0}$ masses of n different stars. By the superposition principle, the gravitational forces exerted on a star at position $\hat{x}$ result from adding the individual forces, i.e., we have

$$\hat{f} = \gamma \hat{m} \sum_{j=1}^{n} m_j \frac{x_j - \hat{x}}{\|x_j - \hat{x}\|_2^3}.$$

Now the evaluation of the force requires $\Theta(n)$ operations. Since even a small galaxy contains around 10^9 and the Milky Way is estimated to contain between 2×10^{11} and 4×10^{11} stars, evaluating $\hat{f}$ takes a long time. If we wanted to simulate the motion of all stars in the Milky Way, we would have to evaluate the forces exerted on each star by all other stars, leading to at least 2×10^{22} operations. This is an amount of computational work that even modern parallel computers cannot handle: a single core of a processor might be able to compute 10^9 forces per second, so it would take at least 2×10^{13} seconds, i.e., approximately 633 761 years. Even if we could parallelize the evaluation perfectly and had 633 761 processor cores at our disposal, one evaluation of the forces would still take a year, and standard time-stepping methods require a large number of evaluations to obtain a reasonably accurate simulation.

We can work around this problem by employing an approximation: let $s \subseteq \mathbb{R}^3$ be a convex subdomain, and let

$$\hat{s} := \{j \in \mathcal{I} \ : \ x_j \in s\}, \qquad \qquad \mathcal{I} := [1 : n]$$

denote the indices of all stars located in s.

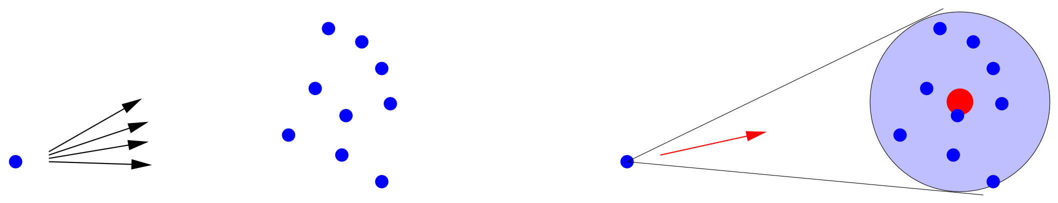

Figure 1.1: Replacing a cluster of stars by a “virtual star” that exerts approximately the same gravitational force

If the diameter of s is small compared to the distance from s to $\hat{x}$, we have

$$\frac{x_j - \hat{x}}{\|x_j - \hat{x}\|_2^3} \approx \frac{x_s - \hat{x}}{\|x_s - \hat{x}\|_2^3} \qquad \text{for all } j \in \hat{s},$$

where x_s denotes the (suitably defined) center of s. Using this approximation, we obtain

$$\sum_{j\in\hat{s}} m_j \frac{x_j - \hat{x}}{\|x_j - \hat{x}\|_2^3} \approx \sum_{j\in\hat{s}} m_j \frac{x_s - \hat{x}}{\|x_s - \hat{x}\|_2^3} = \underbrace{\left(\sum_{j\in\hat{s}} m_j\right)}_{=:m_s} \frac{x_s - \hat{x}}{\|x_s - \hat{x}\|_2^3} = m_s \frac{x_s - \hat{x}}{\|x_s - \hat{x}\|_2^3}.$$

If we have x_s and m_s at our disposal, evaluating this expression requires only $\mathcal{O}(1)$ operations. Essentially we approximate all stars in the region s by a single “virtual” star at position x_s of mass m_s. Considering that s may contain millions or even billions of stars, replacing them by a single one significantly reduces the computational work.

In most cases, a single subdomain s cannot contain all stars and at the same time be sufficiently far from $\hat{x}$. This problem can be solved by using a nested hierarchy of subdomains: given a subdomain s containing a number of stars, we check whether it is sufficiently far from $\hat{x}$. If it is, we use our approximation. Otherwise, we split s into subdomains and check these subdomains recursively. We can arrange the splitting algorithm in a way that guarantees that the diameters of the subdomains decay exponentially, and prove that $\mathcal{O}(\log n)$ subdomains are sufficient to approximate the force acting at $\hat{x}$.

This approach can be generalized and made more efficient, e.g., we can improve the accuracy by using multiple virtual stars per subdomain s, we can reduce the computational work by constructing virtual stars for s from virtual stars of subdomains s' instead of the original stars. The best methods employ hierarchies of source subdomains s and target subdomains t that are *both* represented by virtual stars: in a first step, the stars in the source subdomains are replaced by virtual stars. In a second step, the interactions between virtual source stars in s and virtual target stars in t are computed. In a final step, the forces acting on the virtual target stars in t are translated back to forces acting on real target stars. This technique can reach almost linear complexity and is known as a *fast multipole method* [22] or as a *symmetric panel clustering method* [30]. Its algebraic counterpart is the $\mathcal{H}^2$*-matrix representation* [27, 8].

A particularly interesting application is the approximation of matrices resulting from the discretization of integral equations or partial differential equations: instead of using k virtual stars to approximate a gravitational field, we use k coefficients to approximate the effect of a submatrix. Considered from an algebraic point of view, this is equivalent to approximating a submatrix by a matrix of rank k, and this kind of approximation can be constructed efficiently for matrices appearing in a large number of practical applications.

Splitting a matrix hierarchically into submatrices that can be approximated by low rank leads to the concept of *hierarchical matrices* (or short *$\mathcal{H}$-matrices*) [23, 26, 25, 21, 24]. These matrices are of particular interest, since they can be used to replace fully populated matrices in a wide range of applications using specialized algorithms for handling matrix products, inverses, or factorizations. This approach leads to very efficient and robust preconditioners for integral equations and elliptic partial differential equations, it can be used to evaluate matrix functions or to solve certain kinds of matrix equations.

Taking the concept of low-rank approximations a step further leads to *$\mathcal{H}^2$-matrices* [27, 8, 6] that handle multiple low-rank blocks simultaneously in order to reduce storage requirements and computational work even further.

Acknowledgements

I would like to thank Daniel Hans, Christina Börst, Jens Liebenau, Jonathan Schilling, Jan-Niclas Thiel, Bennet Carstensen, Jonas Lorenzen, Nils Krütgen, and Nico Wichmann for helpful corrections.

2 One-dimensional model problem

We start with a one-dimensional model problem that shares many important properties with "real" applications, but is sufficiently simple to allow us to analyze both accuracy and complexity without the need for elaborate new tools

2.1 Integral equation and discretization

We consider the integral equation

$$\int_0^1 g(x,y)u(y)\,dy = f(x) \qquad \text{for all } x \in [0,1], \tag{2.1}$$

where the *kernel function* g is given by

$$g(x,y) := \begin{cases} -\log|x-y| & \text{if } x \neq y, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } x,y \in [0,1] \tag{2.2}$$

and the right-hand side f and the solution u are in a suitable function spaces. Here $\log(z)$ denotes the natural logarithm of $z \in \mathbb{R}_{>0}$, i.e., we have $\log(e^z) = z$.

In order to compute an approximation of the solution u, we consider a variational formulation: we choose a subspace $\mathcal{V}$ of $L^2[0,1]$ for both the solution u and the right-hand side f and multiply (2.1) by test functions $v \in \mathcal{V}$ to obtain the following problem:

Find $u \in \mathcal{V}$ such that

$$\int_0^1 v(x) \int_0^1 g(x,y)u(y)\,dy\,dx = \int_0^1 v(x)f(x)\,dx \qquad \text{for all } v \in \mathcal{V}.$$

We will not investigate the appropriate choice of the space $\mathcal{V}$ here, although it is of course important for the existence and uniqueness of solutions.

We are only interested in the Galerkin discretization of the variational formulation: we let $n \in \mathbb{N}$ and introduce piecewise constant basis functions

$$\varphi_i(x) := \begin{cases} 1 & \text{if } x \in [(i-1)/n, i/n] \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i \in [1:n],\ x \in [0,1].$$

Following the standard Galerkin approach, the space $\mathcal{V}$ is replaced by

$$\mathcal{V}_n := \operatorname{span}\{\varphi_i \ :\ i \in [1:n]\}$$

to obtain the following finite-dimensional variational problem:

Find $u_n \in \mathcal{V}_n$ such that

$$\int_0^1 v_n(x) \int_0^1 g(x,y) u_n(y)\, dy\, dx = \int_0^1 v_n(x) f(x)\, dx \qquad \text{for all } v_n \in \mathcal{V}_n.$$

In order to compute u_n, we express it in terms of the coefficient vector $z \in \mathbb{R}^n$ corresponding to our basis, i.e., we have

$$u_n = \sum_{j=1}^n z_j \varphi_j.$$

Testing with basis vectors yields

$$\sum_{j=1}^n z_j \int_0^1 \varphi_i(x) \int_0^1 g(x,y) \varphi_j(y)\, dy\, dx = \int_0^1 \varphi_i(x) f(x)\, dx \qquad \text{for all } i \in [1:n],$$

and this describes an n-dimensional linear system of equations.

Introducing the matrix $G \in \mathbb{R}^{n \times n}$ and the vector $b \in \mathbb{R}^n$ by

$$g_{ij} := \int_0^1 \varphi_i(x) \int_0^1 g(x,y) \varphi_j(y)\, dy\, dx \qquad \text{for all } i,j \in [1:n], \tag{2.3a}$$

$$b_i := \int_0^1 \varphi_i(x) f(x)\, dx \qquad \text{for all } i \in [1:n], \tag{2.3b}$$

we can write this linear system in the compact form

$$Gz = b. \tag{2.4}$$

Since G is an n-dimensional square matrix, we have to store n^2 coefficients. In order to make u_n a reasonably accurate approximation of u, we typically have to choose n relatively large, so that storing n^2 coefficients is unattractive.

Since $g(x,y) > 0$ holds for almost all $x, y \in [0,1]$, all coefficients are non-zero, so standard data structures like sparse matrices or band matrices cannot be applied.

Remark 2.1 (Toeplitz matrix) *G is a* Toeplitz matrix*, i.e., we have*

$$j - i = \ell - k \Longrightarrow a_{ij} = a_{k\ell} \qquad \textit{for all } i,j,k,\ell \in [1:n].$$

Toeplitz matrices can be embedded in circulant matrices*, and circulant matrices can be diagonalized using the discrete Fourier transformation.*

This means that we can use the fast Fourier transformation algorithm [10] to evaluate matrix-vector products in $\mathcal{O}(n \log n)$ operations. Unfortunately, this approach relies heavily on the regular structure of our discretization and is typically not an option for more general problems.

Remark 2.2 (Wavelets) *Another approach is to use wavelet basis functions [13, 9] instead of piecewise constant functions.*

This reduces the absolute value of most matrix elements significantly and thereby allows us to approximate the entire matrix G by a sparse matrix that can be handled very efficiently. This approach can be extended to more general geometries and integral operators [12, 14, 11], but the corresponding algorithm is more complicated than the technique we will focus on here.

2.2 Degenerate approximation

We are looking for a *data-sparse* approximation of the matrix G, i.e., a representation that requires us to store only a relatively small number of coefficients. In the case of the integral equation (2.1), we can take advantage of the fact that the kernel function g is analytic as long as we stay away from the singularity at $x = y$: if we have two intervals $t = [a, b]$ and $s = [c, d]$ with $t \cap s = \emptyset$, the restriction $g|_{t\times s}$ of g to the axis-parallel box $t \times s = [a, b] \times [c, d]$ is an analytic function and can be approximated by polynomials.

For the sake of simplicity, we consider a straightforward Taylor expansion of the function $x \mapsto g(x, y)$ for a fixed $y \in s$. For an order $m \in \mathbb{N}$, the Taylor polynomial centered at $x_t \in t$ is given by

$$\tilde{g}_{t,s}(x, y) := \sum_{\nu=0}^{m-1} \frac{(x - x_t)^\nu}{\nu!} \frac{\partial^\nu g}{\partial x^\nu}(x_t, y) \qquad \text{for all } x \in t,\ y \in s. \tag{2.5}$$

This is an example of a *degenerate approximation*, i.e., it is a sum of tensor products

$$\tilde{g}_{ts}(x, y) = \sum_{\nu=0}^{m-1} a_{ts,\nu}(x) b_{ts,\nu}(y) \qquad \text{for all } x \in t,\ y \in s \tag{2.6}$$

with

$$a_{ts,\nu}(x) := \frac{(x - x_t)^\nu}{\nu!}, \qquad b_{ts,\nu}(y) := \frac{\partial^\nu g}{\partial x^\nu}(x_t, y) \qquad \text{for all } \nu \in [0 : m],\ x \in t,\ y \in s.$$

Degenerate approximations are useful because they immediately lead to data-sparse approximations: if we assume $\tilde{g}_{ts} \approx g|_{t\times s}$, we find

$$\begin{aligned} g_{ij} &= \int_0^1 \varphi_i(x) \int_0^1 g(x, y)\varphi_j(y)\, dy\, dx = \int_{(i-1)/n}^{i/n} \int_{(j-1)/n}^{j/n} g(x, y)\, dy\, dx \\ &\approx \int_{(i-1)/n}^{i/n} \int_{(j-1)/n}^{j/n} \tilde{g}_{ts}(x, y)\, dy\, dx = \sum_{\nu=0}^{m-1} \int_{(i-1)/n}^{i/n} a_{ts,\nu}(x)\, dx \int_{(j-1)/n}^{j/n} b_{ts,\nu}(y)\, dy \end{aligned}$$

for all $i, j \in [1 : n]$ with

$$\operatorname{supp} \varphi_i = [(i - 1)/n, i/n] \subseteq t, \qquad \operatorname{supp} \varphi_j = [(j - 1)/n, j/n] \subseteq s. \tag{2.7}$$

We collect the indices of all row basis functions satisfying the first condition in a set

$$\hat{t} := \{i \in [1 : n] \ :\ \operatorname{supp} \varphi_i \subseteq t\},$$

and the indices of all column basis functions satisfying the second condition in

$$\hat{s} := \{j \in [1 : n] \ :\ \operatorname{supp} \varphi_j \subseteq s\},$$

therefore we should be able to replace g by $\tilde{g}_{ts}$ in the integral (2.3a) for all row indices $i \in \hat{t}$ and all column indices $j \in \hat{s}$.

We obtain the approximation

$$g_{ij} \approx \sum_{\nu=0}^{m-1} a_{ts,i\nu} b_{ts,j\nu} \qquad \text{for all } i \in \hat{t},\ j \in \hat{s}$$

with the matrices matrices $A_{ts} \in \mathbb{R}^{\hat{t}\times M}$, $B_{ts} \in \mathbb{R}^{\hat{s}\times M}$, $M := [0 : m-1]$ given by

$$a_{ts,i\nu} := \int_0^1 \varphi_i(x) a_{ts,\nu}(x)\,dx = \int_{(i-1)/n}^{i/n} a_{ts,\nu}(x)\,dx \qquad \text{for all } i \in \hat{t},\ \nu \in M,$$

$$b_{ts,j\nu} := \int_0^1 \varphi_j(y) b_{ts,\nu}(y)\,dy = \int_{(j-1)/n}^{j/n} b_{ts,\nu}(y)\,dy \qquad \text{for all } j \in \hat{s},\ \nu \in M.$$

It is usually more convenient to write this approximation in the short form

$$G|_{\hat{t}\times\hat{s}} \approx A_{ts} B_{ts}^*, \tag{2.8}$$

where B_{ts}^* denotes the adjoint (in this case the transposed) matrix of B_{ts}. If we store the factors A_{ts} and B_{ts} instead of $G|_{\hat{t}\times\hat{s}}$, we only require $m(|\hat{t}| + |\hat{s}|)$ coefficients instead of $|\hat{t} \times \hat{s}| = |\hat{t}|\,|\hat{s}|$. If m is small, the factorized approximation can therefore be significantly more efficient than the original submatrix.

The range of the approximation $A_{ts}B_{ts}^*$ is contained in the range of A_{ts} and therefore at most m-dimensional, therefore the approximation has at most a rank of m. *Factorized low-rank approximations* of this kind are a very versatile tool for dealing with non-local operators and play a crucial role in this book.

2.3 Error analysis

Since we are using an approximation of the kernel function g, we have to investigate the corresponding approximation error.

Reminder 2.3 (Taylor expansion) *Let $z_0, z \in [a,b]$, and let $f \in C^m[a,b]$. We have*

$$f(z) = \sum_{\nu=0}^{m-1} \frac{(z-z_0)^\nu}{\nu!} \frac{\partial^\nu f}{\partial z^\nu}(z_0) + \int_0^1 \frac{(1-\tau)^{m-1}}{(m-1)!} \frac{\partial^m f}{\partial z^m}(z_0 + \tau(z-z_0))\,d\tau\,(z-z_0)^m.$$

Proof. Induction. The base case $m = 1$ is the fundamental theorem of calculus. The induction step is partial integration. ∎

In the case of our approximation of the kernel function g, the statement of Reminder 2.3 takes the form

$$g(x,y) - \tilde{g}_{ts}(x,y) = \int_0^1 \frac{(1-\tau)^{m-1}}{(m-1)!} \frac{\partial^m g}{\partial x^m}(x_t + \tau(x-x_t), y)\,d\tau\,(x-x_t)^m.$$

In order to obtain a useful error estimate, we require the derivatives of g. These are easily obtained, at least for our model problem.

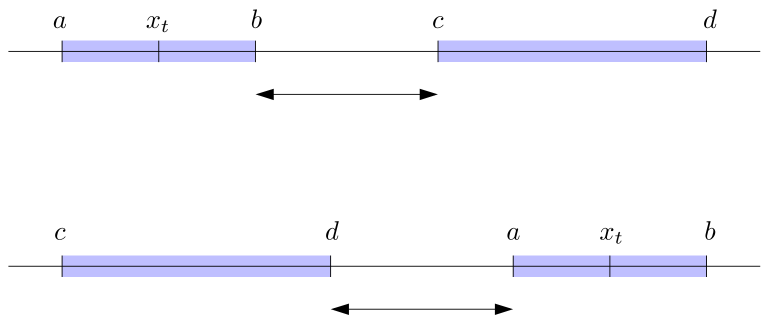

Figure 2.1: Distances and diameters of intervals

Lemma 2.4 *Let $x, y \in \mathbb{R}$ with $x \neq y$. We have*

$$\frac{\partial^m g}{\partial x^m}(x,y) = (-1)^m\,(m-1)!\,(x-y)^{-m},$$
$$\frac{\partial^m g}{\partial y^m}(x,y) = (m-1)!\,(x-y)^{-m} \qquad \text{for all } m \in \mathbb{N}.$$

Proof. Straightforward induction. ■

Since the kernel function g has singularities for $x = y$, the same holds for its derivatives, therefore the chances of bounding the error if the target interval t and the source interval s intersect are looking quite bleak. If we assume that t and s are disjoint, on the other hand, we can not only obtain error bounds, but these bounds even converge exponentially to zero as the order m increases.

Theorem 2.5 (Error estimate) *Let $x_t = (b+a)/2$ denote the midpoint of $t = [a,b]$, let $\operatorname{diam}(t) = b - a$ denote its diameter and $\operatorname{dist}(t,s) = \max\{c-b, a-d, 0\}$ the distance between t and s.*

If $\operatorname{dist}(t,s) > 0$, we have

$$|g(x,y) - \tilde{g}_{ts}(x,y)| \le \log(1+\eta)\left(\frac{\eta}{\eta+1}\right)^{m-1} \qquad \text{for all } x \in t,\ y \in s$$

with the admissibility parameter

$$\eta := \frac{\operatorname{diam}(t)}{2\operatorname{dist}(t,s)}. \tag{2.9}$$

Proof. Let $\operatorname{dist}(t,s) > 0$, and let $x \in t$, $y \in s$. The triangle inequality and the equations for the derivatives provided by Lemma 2.4 yield

$$\begin{aligned}|g(x,y) - \tilde{g}_{ts}(x,y)| &\le \int_0^1 \frac{(1-\tau)^{m-1}}{(m-1)!}\left|\frac{\partial^m g}{\partial x^m}(x_t + \tau(x-x_t), y)\right| d\tau |x-x_t|^m \\ &= \int_0^1 \frac{(1-\tau)^{m-1}}{(m-1)!}\frac{(m-1)!|x-x_t|^m}{|x_t+\tau(x-x_t)-y|^m}\,d\tau\end{aligned}$$

$$\leq \int_0^1 (1-\tau)^{m-1} \left(\frac{|x-x_t|}{(|x_t-y| - \tau|x-x_t|)} \right)^m \, d\tau. \tag{2.10}$$

Due to $\operatorname{dist}(t,s) > 0$, we have either $c > b$ or $a > d$. In the first case (top in Figure 2.1), we have $x, x_t \leq b < c \leq y$. In the second case (bottom in Figure 2.1), we have $y \leq d < a \leq x, x_t$. This implies $x_t - y < 0$ if $c > b$ and $x_t - y > 0$ if $a > d$, so we find

$$|x_t - y| = \begin{cases} y - x_t = y - b + b - x_t \geq c - b + \frac{b-a}{2} & \text{if } c > b, \\ x_t - y = x_t - a + a - y \geq \frac{b-a}{2} + a - d & \text{if } a > d, \end{cases}$$

and conclude $|x_t - y| \geq \operatorname{diam}(t)/2 + \operatorname{dist}(t,s)$. Due to $|x - x_t| \leq \operatorname{diam}(t)/2$, we obtain

$$\frac{|x-x_t|}{|x_t-y| - \tau|x-x_t|} \leq \frac{\operatorname{diam}(t)/2}{\operatorname{dist}(t,s) + (1-\tau)\operatorname{diam}(t)/2}.$$

If $\operatorname{diam}(t) = 0$ holds, the proof is complete.

Assuming now $\operatorname{diam}(t) > 0$, we can introduce

$$\zeta := 1/\eta = \frac{2\operatorname{dist}(t,s)}{\operatorname{diam}(t)},$$

and write our estimate as

$$\frac{|x-x_t|}{|x_t-y| - \tau|x-x_t|} \leq \frac{1}{2\operatorname{dist}(t,s)/\operatorname{diam}(t) + 1 - \tau} = \frac{1}{\zeta + 1 - \tau}.$$

The error estimate (2.10) takes the form

$$|g(x,y) - \tilde{g}_{ts}(x,y)| \leq \int_0^1 \frac{(1-\tau)^{m-1}}{(\zeta+1-\tau)^m} \, d\tau = \int_0^1 \frac{\sigma^{m-1}}{(\zeta+\sigma)^m} \, d\sigma.$$

Due to

$$\frac{\sigma}{\zeta+\sigma} \leq \frac{1}{\zeta+1} \qquad \text{for all } \sigma \in [0,1],$$

we arrive at

$$\begin{aligned} |g(x,y) - \tilde{g}_{ts}(x,y)| &\leq \left(\frac{1}{\zeta+1}\right)^{m-1} \int_0^1 \frac{1}{\zeta+\sigma} \, d\sigma = \left(\frac{1}{\zeta+1}\right)^{m-1} (\log(\zeta+1) - \log(\zeta)) \\ &= \left(\frac{1/\zeta}{1+1/\zeta}\right)^{m-1} \log(1+1/\zeta) = \left(\frac{\eta}{1+\eta}\right)^{m-1} \log(1+\eta). \end{aligned}$$

This is the estimate we need. ∎

If $\operatorname{dist}(t,s) > 0$, we have

$$\eta = \frac{\operatorname{diam}(t)}{2\operatorname{dist}(t,s)} < \infty$$

and the Taylor expansion $\tilde{g}_{ts}(x,y)$ converges exponentially at a rate of

$$\frac{\eta}{\eta+1} < 1$$

to $g(x,y)$ for all $x \in t$ and $y \in s$.

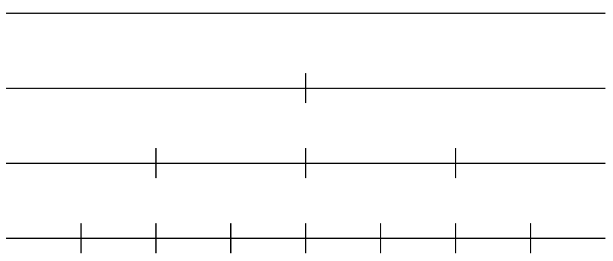

Figure 2.2: Simple cluster tree constructed by recursive bisection of the interval $[0,1]$

2.4 Hierarchical partition

We can expect convergence only if we apply the approximation $\tilde{g}_{ts}$ to subdomains $t \times s$ satisfying $\operatorname{dist}(t,s) > 0$. In order to guarantee a certain rate of convergence, we have to ensure that the parameter η introduced in (2.9) is bounded.

Our approach is to “reverse” the roles of subdomain and admissibility parameter: instead of choosing the paramter to fit the subdomains, we choose the subdomain to fit the parameter.

We fix $\eta \in \mathbb{R}_{>0}$ and check whether a given subdomain $t \times s$ satisfies the *admissibility condition*

$$\operatorname{diam}(t) \leq 2\eta \operatorname{dist}(t,s). \tag{2.11}$$

If it does, we can approximate the kernel function and obtain a factorized low-rank approximation. Otherwise, we split the subdomain, unless it is so small that we can afford storing the corresponding matrix directly.

In this example we use the choice $\eta = 1/2$, i.e., we consider a subdomain $t \times s$ admissible if $\operatorname{diam}(t) \leq \operatorname{dist}(t,s)$ holds. By Theorem 2.5, this leads to the error bound

$$|g(x,y) - \tilde{g}_{ts}(x,y)| \leq 3\log(3/2)\,3^{-m} \qquad \text{for all } x \in t,\ y \in s,\ m \in \mathbb{N}.$$

In general, the parameter η allows us to balance the rate of convergence against the number of subdomains, i.e., accuracy against computational work and storage.

For inadmissible subdomains, we use a simple splitting strategy based on bisection: assume that $t = [a,b]$ and $s = [c,d]$ are *inadmissible*, i.e., that the condition (2.11) does *not* hold. We let

$$a_1 := a, \qquad b_1 = a_2 := \frac{b+a}{2}, \qquad b_2 := b,$$
$$c_1 := c, \qquad d_1 = c_2 := \frac{d+c}{2}, \qquad d_2 := d$$

and define

$$t_1 := [a_1, b_1], \qquad t_2 := [a_2, b_2], \qquad s_1 := [c_1, d_1], \qquad s_2 := [c_2, d_2],$$

i.e., we split t into two equal halves t_1, t_2, and s into two equal halves s_1, s_2.

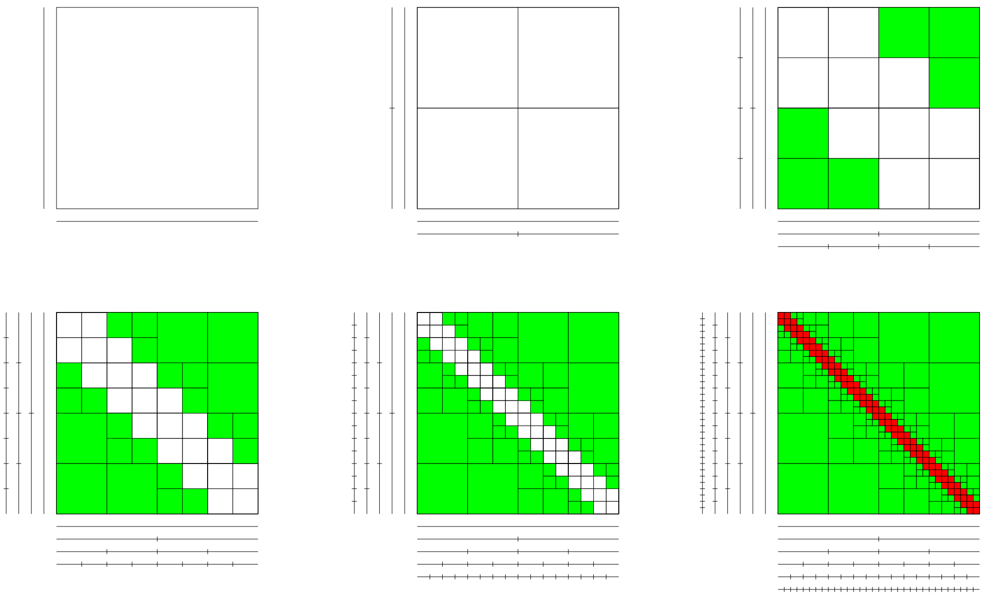

Figure 2.3: Hierarchical subdivision of the domain $[0,1] \times [0,1]$

Now we check whether the Cartesian products $t_1 \times s_1$, $t_1 \times s_2$, $t_2 \times s_1$, and $t_2 \times s_2$ are admissible and proceed by recursion if they are not.

Our construction leads to a subdivision of $[0,1]$ into subintervals of the form

$$t_{\ell,\alpha} := [(\alpha-1)2^{-\ell}, \alpha 2^{-\ell}] \qquad\qquad \text{for all } \ell \in \mathbb{N}_0,\ \alpha \in [1:2^\ell].$$

Each interval $t_{\ell,\alpha}$ is split into

$$t_{\ell+1,2\alpha-1} = [(\alpha-1)2^{-\ell}, (\alpha-1/2)2^{-\ell}] \qquad \text{and} \qquad t_{\ell+1,2\alpha} = [(\alpha-1/2)2^{-\ell}, \alpha 2^{-\ell}].$$

We call these subintervals the *sons* of $t_{\ell,\alpha}$ and arrive at a tree structure, cf. Figure 2.2, describing the subdivision of $[0,1] = t_{0,1}$. The Cartesian product $[0,1] \times [0,1]$ is split into Cartesian products $t \times s$ of pairs of elements of this tree.

Since we only split the domain, there are always subdomains $t \times s$ that include the diagonal $\{(x,y) \in [0,1] \times [0,1] \ : \ x = y\}$ and therefore do not satisfy the admissibility condition (2.11). In order to handle these subdomains, we stop splitting at a given maximal depth $p \in \mathbb{N}_0$ of the recursion and store the remaining matrix entries directly. The resulting decomposition of $[0,1] \times [0,1]$ into admissible and inadmissible subdomains can be seen in Figure 2.3.

The next step is to construct the approximation of the matrix G, i.e., to integrate the products of basis functions and approximated kernel functions. In order to keep this task as simple as possible, we assume that $n = 2^q$ holds with $q \in \mathbb{N}_0$, $q \geq p$. This property guarantees that the support $[(i-1)/n, i/n]$ of a basis function φ_i is either completely contained in one of our subintervals t or that the intersection is a null set, so that the integral vanishes.

Under these conditions, we have

$$\hat{t}_{\ell,\alpha} = [(\alpha-1)2^{q-\ell}+1 : \alpha 2^{q-\ell}],$$
$$t_{\ell,\alpha} = \bigcup\{\operatorname{supp}\varphi_i \ : \ i \in \hat{t}_{\ell,\alpha}\} \qquad \text{for all } \ell \in [0:p],\ \alpha \in [1:2^\ell].$$

Storing the submatrices $G|_{\hat{t}\times\hat{s}}$ for inadmissible domains $t \times s$ on level $\ell = p$ requires $|\hat{t}|\,|\hat{s}| = 4^{q-p}$ coefficients, since $|\hat{t}| = |\hat{s}| = 2^{q-p}$. As long as q is not significantly larger than p, this amount of storage is acceptable.

For admissible domains $t \times s$ on level ℓ, we use the approximation (2.8) and store $|\hat{t}|\,m = 2^{q-\ell}m$ coefficients for the matrix A_{ts} and $|\hat{s}|\,m = 2^{q-\ell}m$ coefficients for the matrix B_{ts}.

Remark 2.6 (Rank) *If we replace g by $\tilde{g}_{ts}$, we obtain an approximation $\widetilde{G} \in \mathbb{R}^{n\times n}$ of the matrix G. Solving $\widetilde{G}\widetilde{x} = b$ instead of $Gx = b$ leads to an error of*

$$\frac{\|x-\widetilde{x}\|}{\|x\|} \le \frac{\kappa(G)}{1-\kappa(G)\frac{\|G-\widetilde{G}\|}{\|G\|}} \frac{\|G-\widetilde{G}\|}{\|G\|}.$$

In typical situations, we expect the condition number $\kappa(G)$ to grow like n^c for a constant $c > 0$, while the discretization error converges like n^{-d} for a constant $d > 0$.

In order to ensure that our approximation of the matrix adds only an additional error on the same order as the discretization error, we need the relative matrix error $\frac{\|G-\widetilde{G}\|}{\|G\|}$ to converge like n^{-c-d}. Due to our admissibility condition and Theorem 2.5, an order of $m \sim \log(n)$ is sufficient to guarantee this property.

2.5 Complexity

Let us now consider the storage requirements of our approximation of the matrix G. If $t \times s$ is admissible, we store the matrices A_{ts} and B_{ts}, and we have already seen that this requires $m(|\hat{t}| + |\hat{s}|)$ coefficients. If $t \times s$ is not admissible, we store $G|_{\hat{t}\times\hat{s}}$ in the standard way, and this requires 4^{q-p} coefficients. In order to obtain an estimate for the approximation of the entire matrix, we have to know how many admissible and inadmissible domains appear in our decomposition of the domain $[0,1] \times [0,1]$.

In our model situation, the construction of the domains is very regular and all domains $t \times s$ fall into one of the following for categories: we call $t \times s$

- *diagonal* if $t = s$,
- *right-neighbouring* if $\max t = \min s$,
- *left-neighbouring* if $\min t = \max s$, and
- *admissible* otherwise.

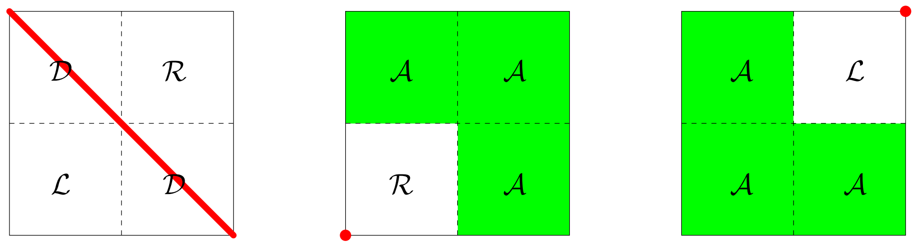

Figure 2.4: Diagonal, right-neighbouring, and left-neighbouring inadmissible domains with corresponding splitting patterns

By our construction, cf. Figure 2.4, a diagonal subdomain $t \times s$ is split into two diagonal subdomains $t_1 \times s_1$ and $t_2 \times s_2$, one right-neighbouring subdomain $t_1 \times s_2$ and one left-neighbouring subdomain $t_2 \times s_1$.

A right-neighbouring subdomain $t \times s$ is split into a right-neighbouring subdomain $t_2 \times s_1$ and three admissible subdomains $t_2 \times s_2$, $t_1 \times s_1$ and $t_1 \times s_2$.

A left-neighbouring subdomain $t \times s$ is split into a left-neighbouring subdomain $t_1 \times s_2$ and three admissible subdomains $t_1 \times s_1$, $t_2 \times s_1$ and $t_2 \times s_2$.

Admissible subdomains are not split.

We can collect the subdomains of one of the four types on each level: for all $\ell \in [0 : p]$, we define

$$\mathcal{D}_\ell := \begin{cases} \{[0,1] \times [0,1]\} & \text{if } \ell = 0, \\ \{t_1 \times s_1, t_2 \times s_2 \ : \ t \times s \in \mathcal{D}_{\ell-1}\} & \text{otherwise,} \end{cases}$$

$$\mathcal{R}_\ell := \begin{cases} \emptyset & \text{if } \ell = 0, \\ \{t_1 \times s_2 \ : \ t \times s \in \mathcal{D}_{\ell-1}\} & \\ \quad \cup \{t_2 \times s_1 \ : \ t \times s \in \mathcal{R}_{\ell-1}\} & \text{otherwise,} \end{cases}$$

$$\mathcal{L}_\ell := \begin{cases} \emptyset & \text{if } \ell = 0, \\ \{t_2 \times s_1 \ : \ t \times s \in \mathcal{D}_{\ell-1}\} & \\ \quad \cup \{t_1 \times s_2 \ : \ t \times s \in \mathcal{L}_{\ell-1}\} & \text{otherwise,} \end{cases}$$

$$\mathcal{A}_\ell := \begin{cases} \emptyset & \text{if } \ell = 0, \\ \{t_2 \times s_2, t_1 \times s_1, t_1 \times s_2 \ : \ t \times s \in \mathcal{R}_{\ell-1}\} & \\ \quad \cup \{t_1 \times s_1, t_2 \times s_1, t_2 \times s_2 \ : \ t \times s \in \mathcal{L}_{\ell-1}\} & \text{otherwise.} \end{cases}$$

These definitions lead to a recurrence relation for the cardinalities of the sets.

Lemma 2.7 (Cardinalities) *We have*

$$|\mathcal{D}_\ell| = 2^\ell, \qquad |\mathcal{R}_\ell| = 2^\ell - 1, \qquad |\mathcal{L}_\ell| = 2^\ell - 1,$$
$$|\mathcal{A}_\ell| = \begin{cases} 0 & \textit{if } \ell = 0, \\ 6(2^{\ell-1} - 1) & \textit{otherwise} \end{cases} \qquad \textit{for all } \ell \in [0 : p].$$

Proof. By induction.

For $\ell = 0$, the equations follow directly from our definitions.

Assume now that the equations hold for $\ell \in [0 : p-1]$. By definition, we have

$$\begin{aligned}
|\mathcal{D}_{\ell+1}| &= 2|\mathcal{D}_\ell| = 2 \cdot 2^\ell = 2^{\ell+1},\\
|\mathcal{L}_{\ell+1}| &= |\mathcal{D}_\ell| + |\mathcal{L}_\ell| = 2^\ell + 2^\ell - 1 = 2^{\ell+1} - 1,\\
|\mathcal{R}_{\ell+1}| &= |\mathcal{D}_\ell| + |\mathcal{R}_\ell| = 2^\ell + 2^\ell - 1 = 2^{\ell+1} - 1,\\
|\mathcal{A}_{\ell+1}| &= 3|\mathcal{L}_\ell| + 3|\mathcal{R}_\ell| = 6(2^\ell - 1).
\end{aligned}$$

This completes the induction. ∎

Theorem 2.8 (Storage requirements) *Let $n = 2^q$, and let p denote the depth of the cluster tree. The approximation of the matrix G requires*

$$6m(p-2)n + (3n + 12m - 2^{q-p+1})2^{q-p} \text{ coefficients.}$$

Proof. The number of coefficients required for the admissible subdomains is

$$\begin{aligned}
\sum_{\ell=0}^{p} \sum_{t\times s\in\mathcal{A}_\ell} (|\hat{t}| + |\hat{s}|)m &= \sum_{\ell=1}^{p} 6(2^{\ell-1} - 1)(2^{q-\ell} + 2^{q-\ell})m = \sum_{\ell=1}^{p} 12(2^{\ell-1} - 1)2^{q-\ell}m\\
&= \sum_{\ell=1}^{p} 12(2^{q-1} - 2^{q-\ell})m = 12m\sum_{\ell=1}^{p} 2^{q-1} - 12m\sum_{\ell=1}^{p} 2^{q-\ell}\\
&= 6mp2^q - 12m2^{q-p}\sum_{\ell=1}^{p} 2^{p-\ell} = 6mpn - 12m2^{q-p}(2^p - 1)\\
&= 6mpn - 12m2^q + 12m2^{q-p} = 6mpn - 12mn + 12m2^{q-p}\\
&= 6m(p-2)n + 12m2^{q-p}.
\end{aligned}$$

The inadmissible subdomains require

$$\begin{aligned}
\sum_{t\times s\in\mathcal{D}_p\cup\mathcal{L}_p\cup\mathcal{R}_p} |\hat{t}|\,|\hat{s}| &= (|\mathcal{D}_p| + |\mathcal{L}_p| + |\mathcal{R}_p|)4^{q-p}\\
&= (2^p + 2^p + 2^p - 2)4^{q-p} = (3\cdot 2^p - 2)4^{q-p}\\
&= 3\cdot 2^q 2^{q-p} - 2^{q-p+1}2^{q-p} = (3n - 2^{q-p+1})2^{q-p}
\end{aligned}$$

coefficients. Adding both results yields the required equation. ∎

Although exact, the result of Theorem 2.8 is not particularly instructive.

We can obtain a more convenient upper bound if we assume that the order m is not too high compared to the number of basis functions n and that we subdivide clusters only as long as they contain more than $2m$ indices. The second assumption can be guaranteed by stopping the splitting process at the right time. The first assumption is manageable since Remark 2.6 leads us to expect $m \sim \log(n)$, i.e., for even moderately-sized matrix dimensions n, the order m should be far smaller than n.

Corollary 2.9 (Storage requirements) *Let $4m \leq n$ and $m < 2^{q-p} \leq 2m$.*
Then our approximation requires less than $6mpn$ coefficients.

Proof. The estimate provided by Theorem 2.8 gives rise to the upper bound

$$\begin{aligned} 6m(p-2)n + (3n + 12m - 2^{q-p+1})2^{q-p} &< 6m(p-2)n + (3n + 12m - 2m)2m \\ &= 6m(p-2)n + 6mn + 20m^2 \\ &< 6m(p-2)n + 6mn + 24m^2 \\ &\leq 6m(p-2)n + 6mn + 6mn = 6mpn \end{aligned}$$

for the number of coefficients. ■

Remark 2.10 (Setup) *In order to construct the matrices A_{ts} and B_{ts} of our approximation, we can take advantage of the fact that the recurrence equations*

$$a_{ts,\nu}(x) = \begin{cases} 1 & \text{if } \nu = 0, \\ \frac{x-x_t}{\nu} a_{ts,\nu-1}(x) & \text{otherwise}, \end{cases} \qquad \text{for all } x \in t,\ \nu \in [0:m],$$

$$b_{ts,\nu}(y) = \begin{cases} -\log|x_t - y| & \text{if } \nu = 0, \\ -\frac{1}{x_t-y} & \text{if } \nu = 1, \\ -\frac{\nu-1}{x_t-y} b_{ts,\nu-1}(y) & \text{otherwise} \end{cases} \qquad \text{for all } y \in s,\ \nu \in [0:m]$$

allow us to evaluate $a_{ts,\nu}$ and $b_{ts,\nu}$ very efficiently. Since $a_{ts,\nu+1}$ is an antiderivative of $a_{ts,\nu}$ and $b_{ts,\nu-1}$ is an antiderivative of $-b_{ts,\nu}$, all integrals appearing in A_{ts} and B_{ts}, and therefore all coefficients, can be computed in $\mathcal{O}(m(|\hat{t}| + |\hat{s}|))$ operations. In particular, we need only $\mathcal{O}(1)$ operations per coefficient.

For the inadmissible domains, we can compute the coefficients g_{ij} by using a second antiderivative of g.

In total we require $\mathcal{O}(1)$ operations for each coefficient, and Corollary 2.9 yields that we only require $\mathcal{O}(mpn)$ operations to set up the entire matrix approximation.

Remark 2.11 (Matrix-vector multiplication) *Once we have constructed the approximation of G, multiplying it by a vector $x \in \mathbb{R}^n$ and adding the result to $y \in \mathbb{R}^n$ is straightforward: for each subdomain $t \times s$, we multiply $x|_{\hat{s}}$ by $A_{ts}B^*_{ts}$ and add the result to $y|_{\hat{t}}$. If we first compute the auxiliary vector $z = B^*_{ts}x|_{\hat{s}}$ and then $A_{ts}z$, we only require $2m(|\hat{t}| + |\hat{s}|)$ operations.*

In order to obtain a bound for the total complexity, we note that for each stored coefficient exactly one multiplication and one addition are carried out. Corollary 2.9 immediately yields that less than

$$12mpn \text{ operations}$$

are required for the complete matrix-vector multiplication: our approximation not only saves storage, but also time.

2.6 Approximation by interpolation

For more general kernel functions g, it may be challenging to find useable equations for the derivatives required by the Taylor expansion and to come up with an efficient algorithm for computing the integrals determining the coefficients of A_{ts} and B_{ts}.

In these situations, *interpolation* offers an elegant and practical alternative: we choose a degree $m \in \mathbb{N}_0$ and denote the set of polynomials of degree m by

$$\Pi_m := \left\{ x \mapsto \sum_{i=0}^{m} a_i x^i \ : \ a_0, \ldots, a_m \in \mathbb{R} \right\}.$$

Given a function $f \in C[a,b]$ and distinct interpolation points $\xi_0, \ldots, \xi_m \in [a,b]$, we look for a polynomial $p \in \Pi_m$ satisfying the equations

$$p(\xi_i) = f(\xi_i) \qquad \text{for all } i \in [0:m]. \tag{2.12}$$

Under suitable conditions, p is a good approximation of f, and interpolation can be used to obtain a degenerate approximation of the kernel function g.

Lemma 2.12 (Lagrange polynomials) *The* Lagrange polynomials *for the distinct interpolation points $\xi_0, \ldots, \xi_m \in [a,b]$ are given by*

$$\ell_\nu(x) := \prod_{\substack{\mu=0 \\ \mu\neq\nu}}^{m} \frac{x - \xi_\mu}{\xi_\nu - \xi_\mu} \qquad \textit{for all } x \in \mathbb{C},\ \nu \in [0:m]. \tag{2.13}$$

All of these polynomials are elements of Π_m and satisfy

$$\ell_\nu(\xi_\mu) = \begin{cases} 1 & \textit{if } \nu = \mu, \\ 0 & \textit{otherwise} \end{cases} \qquad \textit{for all } \nu, \mu \in [0:m]. \tag{2.14}$$

Proof. As products of m linear factors, the Lagrange polynomials are in Π_m.

Let $\nu, \mu \in [0:m]$. If $\nu = \mu$, all factors in (2.13) are equal to one, and so is the product. If $\nu \neq \mu$, one of the factors is equal to zero and the product vanishes. ■

Using Lagrange polynomials, the interpolating polynomial of (2.12) takes the form

$$p = \sum_{\nu=0}^{m} f(\xi_\nu)\ell_\nu. \tag{2.15}$$

Applying this equation to $x \mapsto g(x,y)$ with a fixed $y \in [c,d]$ yields

$$\tilde{g}_{ts}(x,y) := \sum_{\nu=0}^{m} \ell_\nu(x) g(\xi_\nu, y),$$

and this is again a degenerate approximation of g, but it requires us only to be able to evaluate g, not its derivatives.

Remark 2.13 (Setup) *Using interpolation to approximate g leads to the matrices $A_{ts} \in \mathbb{R}^{\hat{t}\times M}$ and $B_{ts} \in \mathbb{R}^{\hat{s}\times M}$, $M = [0:m]$, with entries*

$$a_{ts,i\nu} = \int_0^1 \varphi_i(x)\ell_\nu(x)\,dx, \tag{2.16}$$

$$b_{ts,j\nu} = \int_0^1 \varphi_j(y)g(\xi_\nu, y)\,dy \qquad \textit{for all } i \in \hat{t},\ j \in \hat{s},\ \nu \in M. \tag{2.17}$$

The first integral can be computed by using a quadrature rule that is exact for polynomials of degree m.

For the second integral, we can use the antiderivative in simple situations like the model problem or also rely on quadrature, since the function $y \mapsto g(\xi_\nu, y)$ is smooth: the admissibility condition guarantees that $\xi_\nu \in t$ is sufficiently far from $y \in s$.

In order to construct an approximation of the entire matrix G by interpolation, we require interpolation points for all intervals t appearing in our decomposition of the domain $[0,1] \times [0,1]$.

It is a good idea to start with interpolation points in a fixed reference interval and transform them to all the other intervals, since this approach allows us to obtain uniform error estimates for all subdomains.

We choose the reference interval $[-1,1]$ and interpolation points $\xi_0, \dots \xi_m \in [-1,1]$. Inspired by (2.15), we define an operator that maps a function f to its interpolating polynomial.

Definition 2.14 (Interpolation operator) *The linear operator*

$$\mathfrak{I} \colon C[-1,1] \to \Pi_m, \qquad f \mapsto \sum_{\nu=0}^{m} f(\xi_\nu)\ell_\nu,$$

is called the interpolation operator *for the interpolation points $\xi_0, \dots, \xi_m$.*

In order to obtain interpolation operators for a general interval $[a,b]$, we use the bijective linear mapping

$$\Phi_{[a,b]} \colon \mathbb{C} \to \mathbb{C}, \qquad \hat{x} \mapsto \frac{b+a}{2} + \frac{b-a}{2}\hat{x},$$

that maps $[-1,1]$ to $[a,b]$ and can be used to turn a function $f \in C[a,b]$ into a function $\hat{f} := f \circ \Phi_{[a,b]} \in C[-1,1]$. Interpolating this function and using $\Phi_{[a,b]}^{-1}$ to map the result back to $[a,b]$ yields the *transformed interpolation operator*

$$\mathfrak{I}_t \colon C[a,b] \to \Pi_m, \qquad f \mapsto \mathfrak{I}[f \circ \Phi_{[a,b]}] \circ \Phi_{[a,b]}^{-1}.$$

For our purposes, it would be very useful to be able to represent the transformed interpolation operator $\mathfrak{I}_t$ in the same form as $\mathfrak{I}$ using suitable interpolation points and Lagrange polynomials.

Reminder 2.15 (Identity theorem) *Let $p \in \Pi_m$. If $p(\xi_\nu) = 0$ holds for $m+1$ distinct points $\xi_0, \ldots, \xi_m \in \mathbb{R}$, we have $p = 0$.*

Proof. We consider the linear mapping

$$\Psi\colon \Pi_m \to \mathbb{R}^{m+1}, \qquad p \mapsto \begin{pmatrix} p(\xi_0) \\ \vdots \\ p(\xi_m) \end{pmatrix}.$$

Due to Lemma 2.12, Ψ is surjective, i.e., its rank is $m+1$. Our definition implies $\dim \Pi_m \leq m+1$, and the rank-nullity theorem yields that Ψ is injective. ∎

Lemma 2.16 (Transformed interpolation) *We define the transformed interpolation points and corresponding Lagrange polynomials*

$$\xi_{t,\nu} := \Phi_{[a,b]}(\xi_\nu), \qquad \ell_{t,\nu}(x) := \prod_{\substack{\mu=0 \\ \mu\neq\nu}}^{m} \frac{x - \xi_{t,\mu}}{\xi_{t,\nu} - \xi_{t,\mu}} \qquad \text{for all } \nu \in [0:m],\ x \in \mathbb{C}.$$

We have $\ell_{t,\nu} = \ell_\nu \circ \Phi_{[a,b]}^{-1}$ for all $\nu \in [0:m]$ and therefore

$$\mathfrak{I}_t[f] = \sum_{\nu=0}^{m} f(\xi_{t,\nu}) \ell_{t,\nu} \qquad \text{for all } f \in C[a,b].$$

Proof. Let $\nu \in [0:m]$. Since $\Phi_{[a,b]}$ is bijective, it suffices to prove $\ell_{t,\nu} \circ \Phi_{[a,b]} = \ell_\nu$.
We have

$$\ell_{t,\nu} \circ \Phi_{[a,b]}(\xi_\mu) = \ell_{t,\nu}(\xi_{t,\mu}) = \begin{cases} 1 & \text{if } \nu = \mu, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } \mu \in [0:m].$$

This means that $\ell_{t,\nu} \circ \Phi_{[a,b]}$ and $\ell_\nu \in \Pi_m$ take identical values in the distinct points $\xi_0, \ldots, \xi_m$. Since both are polynomials of degree m, the identity theorem (cf. Reminder 2.15) yields $\ell_{t,\nu} \circ \Phi_{[a,b]} = \ell_\nu$. ∎

Now we can proceed as in the case of the Taylor expansion: we apply interpolation to the function $x \mapsto g(x,y)$ for a fixed $y \in s$ to find

$$g(x,y) \approx \tilde{g}_{ts}(x,y) := \sum_{\nu=0}^{m} \ell_{t,\nu}(x) g(\xi_{t,\nu}, y) \qquad \text{for all } x \in t,\ y \in s. \tag{2.18}$$

Reminder 2.17 (Chebyshev interpolation) *Using the* Chebyshev points

$$\xi_\nu := \cos\left(\pi \frac{2\nu+1}{2m+2}\right) \qquad \text{for all } \nu \in [0:m], \tag{2.19}$$

for interpolation is particularly attractive, since they lead both to a numerically stable algorithm and good error estimates.

2.7 Interpolation error analysis

Let us now consider the analysis of the interpolation error.

Reminder 2.18 (Interpolation error) *Let $\xi_0, \ldots, \xi_m \in [-1,1]$ be distinct interpolation points and the* node polynomial

$$\omega(x) := \prod_{\nu=0}^{m} x - \xi_\nu \qquad \text{for all } x \in \mathbb{R}. \tag{2.20}$$

Let $f \in C^{m+1}[-1,1]$. For every $x \in [-1,1]$, there is an $\eta \in [-1,1]$ with

$$f(x) - \mathfrak{I}[f](x) = \omega(x) \frac{f^{(m+1)}(\eta)}{(m+1)!}. \tag{2.21}$$

Proof. [31] If $x \in \{\xi_0, \ldots, \xi_m\}$, the equation holds trivially for all $\eta \in [-1,1]$. Otherwise, we have $\omega(x) \neq 0$ and can find $R \in \mathbb{R}$ with $0 = f(x) - \mathfrak{I}[f](x) - R\,\omega(x)$.

This means that the function $g(y) := f(y) - \mathfrak{I}[f](y) - R\,\omega(y)$ vanishes in the $m+2$ distinct points $\xi_0, \ldots, \xi_m, x$, and the mean value theorem of differential calculus yields that there is an $\eta \in [-1,1]$ such that $g^{(m+1)}(\eta) = 0$. Due to $\mathfrak{I}[f] \in \Pi_m$ and $\omega^{(m+1)} = (m+1)!$, this implies the equation. ∎

Using the maximum norm

$$\|f\|_{\infty,[a,b]} := \max\{|f(x)| \;:\; x \in [a,b]\} \qquad \text{for all } f \in C[a,b],$$

we can write (2.21) in the compact form

$$\|f - \mathfrak{I}[p]\|_{\infty,[-1,1]} \leq \|\omega\|_{\infty,[-1,1]} \frac{\|f^{(m+1)}\|_{\infty,[-1,1]}}{(m+1)!}.$$

In order to investigate the transformed interpolation operator, we consider a function $f \in C^{m+1}[a,b]$ and introduce again $\hat{f} := f \circ \Phi_{[a,b]} \in C^{m+1}[-1,1]$ to find

$$\begin{aligned} \|f - \mathfrak{I}_t[f]\|_{\infty,[a,b]} &= \|f \circ \Phi_{[a,b]} - \mathfrak{I}_t[f] \circ \Phi_{[a,b]}\|_{\infty,[-1,1]} \\ &= \|\hat{f} - \mathfrak{I}[\hat{f}]\|_{\infty,[-1,1]} \leq \|\omega\|_{\infty,[-1,1]} \frac{\|\hat{f}^{(m+1)}\|_{\infty,[-1,1]}}{(m+1)!}. \end{aligned}$$

The chain rule yields

$$\hat{f}^{(m+1)} = \left(\frac{b-a}{2}\right)^{m+1} f^{(m+1)} \circ \Phi_{[a,b]}$$

and we conclude

$$\|f - \mathfrak{I}_t[f]\|_{\infty,[a,b]} \leq \|\omega\|_{\infty,[-1,1]} \left(\frac{b-a}{2}\right)^{m+1} \frac{\|f^{(m+1)}\|_{\infty,[a,b]}}{(m+1)!}. \tag{2.22}$$

To apply this result to the kernel function, we fix $y \in s$ and apply the estimate to $f(x) := g(x,y)$ to find

$$|g(x,y) - \tilde{g}_{ts}(x,y)| \leq \|\omega\|_{\infty,[-1,1]} \left(\frac{b-a}{2}\right)^{m+1} \frac{\|f^{(m+1)}\|_{\infty,[a,b]}}{(m+1)!} \qquad \text{for all } x \in t.$$

According to Lemma 2.4, we have

$$|f^{(m+1)}(x)| = \left|\frac{\partial^{m+1} g}{\partial x^{m+1}}(x,y)\right| \leq \frac{m!}{\operatorname{dist}(t,s)^{m+1}} \qquad \text{for all } x \in t,$$

so the interpolation error satisfies

$$\|g - \tilde{g}_{ts}\|_{\infty,t\times s} \leq \frac{\|\omega\|_{\infty,[-1,1]}}{m+1} \left(\frac{\operatorname{diam}(t)}{2\operatorname{dist}(t,s)}\right)^{m+1}.$$

If the admissibility condition (2.11) holds, this estimate takes the form

$$\|g - \tilde{g}_{ts}\|_{\infty,t\times s} \leq \frac{\|\omega\|_{\infty,[-1,1]}}{m+1} \eta^{m+1}.$$

Let us now consider Chebyshev interpolation (cf. Reminder 2.17). Using Chebyshev points implies $\omega(x) = 2^{-m}\cos((m+1)\arccos(x))$ for all $x \in [-1,1]$, and in particular $\|\omega\|_{\infty,[-1,1]} = 2^{-m}$. Our error estimates take the form

$$\|f - \mathfrak{I}_t[f]\|_{\infty,[a,b]} \leq 2\left(\frac{b-a}{4}\right)^{m+1} \frac{\|f^{(m+1)}\|_{\infty,[a,b]}}{(m+1)!} \qquad \text{for all } f \in C^{m+1}[a,b], \tag{2.23a}$$

$$\|g - \tilde{g}_{ts}\|_{\infty,t\times s} \leq \frac{2}{m+1}\left(\frac{\eta}{2}\right)^{m+1}. \tag{2.23b}$$

Compared to Taylor expansion (cf. Theorem 2.5), we can see that the Chebyshev interpolation error converges at a better rate if we have $\eta < 1$, e.g., at a rate of $1/4$ instead of $1/3$ for $\eta = 1/2$.

Even for large values of η, we can prove that Chebyshev interpolation leads to almost the same rate of convergence as Taylor expansion: Chebyshev interpolation is *stable*, i.e., we have

$$\|\mathfrak{I}[f]\|_{\infty,[-1,1]} \leq \Lambda_m \|f\|_{\infty,[-1,1]} \qquad \text{for all } f \in C[-1,1], \tag{2.24}$$

where the stability constant (or *Lebesgue number*) satisfies

$$\Lambda_m \leq \frac{2}{\pi}\log(m+1) + 1 \leq m+1 \qquad \text{for all } m \in \mathbb{N}_0 \tag{2.25}$$

(cf. [29]). The stability estimate (2.24) gives rise to a best-approximation result for Lagrange interpolation.

Lemma 2.19 (Best approximation) *We have*

$$\|f - \mathfrak{I}[f]\|_{\infty,[-1,1]} \leq (1 + \Lambda_m)\|f - p\|_{\infty,[-1,1]} \qquad \text{for all } f \in C[-1,1],\ p \in \Pi_m,$$
$$\|f - \mathfrak{I}_t[f]\|_{\infty,[a,b]} \leq (1 + \Lambda_m)\|f - p\|_{\infty,[a,b]} \qquad \text{for all } f \in C[a,b],\ p \in \Pi_m.$$

Proof. Let $f \in C[-1,1]$ and $p \in \Pi_m$. Due to the identity theorem for polynomials (cf. Reminder 2.15), the interpolation operator is a projection into Π_m, so we have

$$\mathfrak{I}[p] = p. \tag{2.26}$$

Combining the triangle inequality and the stability estimate (2.24) yields

$$\begin{aligned}\|f - \mathfrak{I}[f]\|_{\infty,[-1,1]} &= \|f - p + \mathfrak{I}[p] - \mathfrak{I}[f]\|_{\infty,[-1,1]} = \|f - p + \mathfrak{I}[p-f]\|_{\infty,[-1,1]}\\ &\leq \|f - p\|_{\infty,[-1,1]} + \|\mathfrak{I}[p-f]\|_{\infty,[-1,1]}\\ &\leq \|f - p\|_{\infty,[-1,1]} + \Lambda_m\|p - f\|_{\infty,[-1,1]}.\end{aligned}$$

This is the first estimate.

Let now $f \in C[a,b]$ and $p \in \Pi_m$. We define $\hat{f} := f \circ \Phi_{[a,b]}$ and $\hat{p} := p \circ \Phi_{[a,b]} \in \Pi_m$ and apply the first estimate to obtain

$$\begin{aligned}\|f - \mathfrak{I}_t[f]\|_{\infty,[a,b]} &= \|f \circ \Phi_{[a,b]} - \mathfrak{I}_t[f] \circ \Phi_{[a,b]}\|_{\infty,[-1,1]} = \|\hat{f} - \mathfrak{I}[\hat{f}]\|_{\infty,[-1,1]}\\ &\leq (1 + \Lambda_m)\|\hat{f} - \hat{p}\|_{\infty,[-1,1]} = (1 + \Lambda_m)\|f - p\|_{\infty,[a,b]}.\end{aligned}$$

This is the second estimate. ■

Theorem 2.20 (Interpolation error) *If the admissibility condition (2.11) holds, the interpolation error can be bounded by*

$$\|g - \tilde{g}_{ts}\|_{\infty,t\times s} \leq (1 + \Lambda_m)\log(1+\eta)\left(\frac{\eta}{\eta+1}\right)^m.$$

Proof. Let $y \in s$ and

$$f\colon t \to \mathbb{R}, \qquad x \mapsto g(x,y).$$

Let $p \in \Pi_m$ denote the Taylor polynomial of order m. Due to Theorem 2.5, we have

$$\|f - p\|_{\infty,[a,b]} \leq \log(1+\eta)\left(\frac{\eta}{\eta+1}\right)^m.$$

Now we can apply Lemma 2.19 to conclude

$$\begin{aligned}|g(x,y) - \tilde{g}_{ts}(x,y)| &= |f(x) - \mathfrak{I}_t[f](x)| \leq (1 + \Lambda_m)\|f - p\|_{\infty,[a,b]}\\ &\leq (1 + \Lambda_m)\log(1+\eta)\left(\frac{\eta}{\eta+1}\right)^m \qquad \text{for all } x \in t,\end{aligned}$$

and since y has been chosen arbitrarily, this is already the required result. ■

Due to (2.25), the stability constant Λ_m grows only very slowly as m increases, so the interpolation error converges almost as quickly as the Taylor approximation error.

Remark 2.21 (Chebyshev approximation) *We denote the* Chebyshev polynomials *by*

$$C_m(x) := \cos(m \arccos(x)) \qquad \text{for all } m \in \mathbb{N}_0,\ x \in [-1,1].$$

Since $\{C_0, \dots, C_m\}$ are a basis of Π_m, any polynomial $p \in \Pi_m$ can be expressed as the Chebyshev expansion

$$p(x) = \sum_{\nu=0}^{m} a_\nu C_\nu(x) \qquad \text{for all } x \in [-1,1], \tag{2.27}$$

where $a_0, \dots, a_m \in \mathbb{R}$. This expansion is attractive, since the recurrence relation

$$C_m(x) = \begin{cases} 1 & \text{if } m = 0, \\ x & \text{if } m = 1, \\ 2xC_{m-1}(x) - C_{m-2}(x) & \text{otherwise} \end{cases} \qquad \text{for all } m \in \mathbb{N}_0,\ x \in [-1,1] \tag{2.28}$$

allows us to evaluate (2.27) efficiently in $\mathcal{O}(m)$ operations using the Clenshaw-Curtis algorithm.

Remark 2.22 (Fourier expansion) *Given a function $f \in C[-1,1]$, we can obtain suitable coefficients of a Chebyshev approximation (2.27) via a Fourier expansion.*

Substituting $x = \cos(y)$ yields

$$\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}\,dx = \int_0^\pi f(\cos(y))\,dy,$$

and $\cos(x)\cos(y) = \frac{1}{2}(\cos(x+y) + \cos(x-y))$ yields

$$\int_{-1}^{1} \frac{C_\nu(x) C_\mu(y)}{\sqrt{1-x^2}}\,dx = \begin{cases} \pi & \text{if } \nu = \mu = 0, \\ \pi/2 & \text{if } \nu = \mu > 0, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } \nu, \mu \in \mathbb{N}_0.$$

This orthogonality relation leads to

$$a_0 = \frac{1}{\pi} \int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}} dx, \tag{2.29a}$$

$$a_\nu = \frac{2}{\pi} \int_{-1}^{1} \frac{f(x) C_\nu(x)}{\sqrt{1-x^2}}\,dx \qquad \text{for all } \nu \in [1:m]. \tag{2.29b}$$

Remark 2.23 (Chebyshev interpolation) *With the Chebyshev points defined in (2.19), we have the discrete orthogonality relation*

$$\sum_{\kappa=0}^{m} C_\nu(\xi_\kappa) C_\mu(\xi_\kappa) = \begin{cases} m & \text{if } \nu = \mu = 0, \\ m/2 & \text{if } \nu = \mu > 0, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } \nu, \mu \in [0:m],$$

and we can replace the integrals in (2.29) by sums requiring only $f(\xi_\kappa)$ for $\kappa \in [0:m]$.

In this case, (2.27) coincides with Chebyshev interpolation.

2.8 Improved interpolation error estimates*

We can find significantly better estimates for the interpolation error by following the approach described in [15, Chapter 7, Section 8]: we replace the Taylor expansion by the Chebyshev expansion (2.27).

We focus on the approximation of function $f \in C[-1,1]$, since we can switch to arbitrary intervals using the mapping $\Phi_{[a,b]}$. If the function f can be approximated reasonably well by polynomials of increasing degree, it can be represented by a power series, and therefore also extended to a *holomorphic*, i.e., complex differentiable, function in a neighbourhood of $[-1,1]$.

In this section, we therefore focus on the approximation of functions that are holomorphic in a neighbourhood of the reference interval $[-1,1]$.

To this end, we will represent $[-1,1]$ as the range of the cosine, and by extension as the real part of points on the complex unit circle: Using Euler's formula, we find

$$\cos(t) = \Re(e^{\iota t}) = \frac{e^{\iota t} + \overline{e^{\iota t}}}{2} = \frac{e^{\iota t} + e^{-\iota t}}{2} = \frac{e^{\iota t} + 1/e^{\iota t}}{2} \qquad \text{for all } t \in \mathbb{R},$$

where $\iota \in \mathbb{C}$ denotes the imaginary unit.

Definition 2.24 (Joukowsky transformation) *The mapping*

$$g\colon \mathbb{C} \setminus \{0\} \to \mathbb{C}, \qquad z \mapsto \frac{z + 1/z}{2},$$

is called the Joukowsky transformation.

Complex numbers on the unit circle $\mathcal{S}_1 := \{z \in \mathbb{C} \ : \ |z| = 1\}$ are mapped to their real parts by the Joukowsky transformation, but g is holomorphic, while $z \mapsto \Re(z)$ is not.

Our plan is to obtain an approximation of f by considering the Laurent series of a holomorphic extension of the function

$$\hat{f}\colon \mathcal{S}_1 \to \mathbb{C}, \qquad z \mapsto f(g(z)).$$

In order to define this extension, we have to take a closer look at the Joukowsky transformation.

Lemma 2.25 (Joukowsky transformation) *We have*

$$x < y \iff g(x) < g(y) \qquad \text{for all } x, y \in \mathbb{R}_{\geq 1}. \tag{2.30}$$

Let $\varrho \in \mathbb{R}_{>1}$. The Joukowsky transformation maps the circle $\{z \in \mathbb{C} \ : \ |z| = \varrho\}$ bijectively to the Bernstein ellipse

$$\mathcal{E}_\varrho := \{w \in \mathbb{C} \ : \ |w-1| + |w+1| = 2\alpha\} \tag{2.31}$$

with semi-major axis $\alpha = g(\varrho)$ and foci 1 and -1 (cf. Figure 2.5).

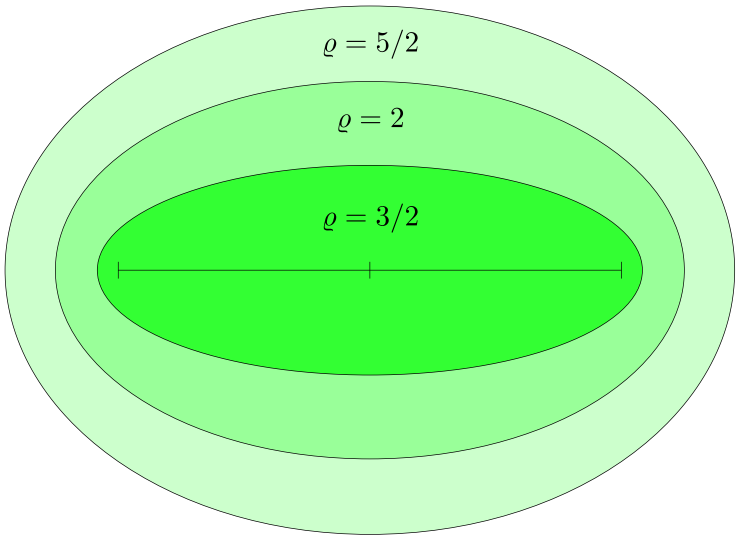

Figure 2.5: Bernstein ellipses for $\varrho \in \{3/2, 2, 5/2\}$.

Proof. Let $x, y \in \mathbb{R}_{\geq 1}$. If we assume $x < y$, we have $xy > 1$ and therefore $\frac{1}{xy} < 1$. This implies

$$\left(1 - \frac{1}{xy}\right) x < \left(1 - \frac{1}{xy}\right) y \iff x - \frac{1}{y} < y - \frac{1}{x}$$
$$\iff x + \frac{1}{x} < y + \frac{1}{y} \iff g(x) < g(y).$$

Assume now $g(x) < g(y)$. This implies $x \neq y$ and therefore again $xy > 1$ and $\frac{1}{xy} < 1$, so we can proceed as before to conclude $x < y$.

Let now $\varrho \in \mathbb{R}_{>1}$. Let $z \in \mathbb{C} \setminus \{0\}$ and $w := g(z)$. We have

$$\begin{aligned}
|w - 1| + |w + 1| = |g(z) - 1| + |g(z) + 1| &= \frac{|z + 1/z - 2|}{2} + \frac{|z + 1/z + 2|}{2} \\
&= \frac{|z^2 + 1 - 2z|}{2|z|} + \frac{|z^2 + 1 + 2z|}{2|z|} = \frac{|z - 1|^2 + |z + 1|^2}{2|z|} \\
&= \frac{(z - 1)\overline{(z - 1)} + (z + 1)\overline{(z + 1)}}{2|z|} \\
&= \frac{(z - 1)(\bar{z} - 1) + (z + 1)(\bar{z} + 1)}{2|z|} \\
&= \frac{|z|^2 - z - \bar{z} + 1 + |z|^2 + z + \bar{z} + 1}{2|z|} \\
&= \frac{2|z|^2 + 2}{2|z|} = |z| + 1/|z|.
\end{aligned}$$

If we have $|z| = \varrho$, this equation immediately implies

$$|w+1| + |w-1| = \varrho + 1/\varrho = 2g(\varrho) = 2\alpha,$$

i.e., $w \in \mathcal{E}_\varrho$.

On the other hand, assume $w \in \mathcal{E}_\varrho$. To find $z \in \mathbb{C} \setminus \{0\}$ with $g(z) = w$, we consider

$$w = g(z) = \frac{z + 1/z}{2} \iff 2wz = z^2 + 1 \iff z^2 - 2wz + 1 = 0. \tag{2.32}$$

This quadratic equation has two solutions $z_1, z_2 \in \mathbb{C}$ with $z_1 z_2 = 1$, i.e., $|z_1|\,|z_2| = 1$. Without loss of generality, we can therefore ensure $|z_1| \geq 1 \geq |z_2|$. Choosing $z := z_1$ guarantees both $w = g(z)$ and $|z| \geq 1$.

Our previous calculation yields

$$2g(|z|) = |z| + 1/|z| = |w+1| + |w-1| = 2\alpha = 2g(\varrho),$$

and due to $\varrho \geq 1$ and $|z| \geq 1$, $g(|z|) = g(\varrho)$ already implies $|z| = \varrho$ via (2.30).

Now we only have to prove that g is injective from $\{z \in \mathbb{C} \ : \ |z| = \varrho\}$ to $\mathcal{E}_\varrho$. We have already seen that we can find $z \in \mathbb{C}$ with $|z| = \varrho$ and $g(z) = w$ for every $w \in \mathcal{E}_\varrho$. If there is another $\tilde{z} \in \mathbb{C}\setminus\{0\}$ with $g(\tilde{z}) = w$, it also has to be a solution of the quadratic equation (2.32). This equation has only two solutions z_1 and z_2, so either we have $\tilde{z} = z_1 = z$ or $\tilde{z} = z_2$. In the latter case, $1 = |z_1|\,|z_2|$ implies $|\tilde{z}| = |z_2| = 1/|z_1| = 1/\varrho < 1$ and therefore $|\tilde{z}| \neq \varrho$ in particular, i.e., z is the only solution of $g(z) = w$ with $|z| = \varrho$. ∎

In order to have $\hat{f} = f \circ g$ holomorphic in a neighbourhood of the unit circle, f has to be holomorphic in a neighbourhood of the reference interval $[-1, 1]$.

Lemma 2.26 (Annulus) *Let $\varrho \in \mathbb{R}_{\geq 1}$. The Joukowsky transformation g maps the annulus*

$$\mathcal{A}_\varrho := \{z \in \mathbb{C} \ : \ 1/\varrho \leq |z| \leq \varrho\}.$$

onto the Bernstein disc

$$\mathcal{D}_\varrho := \{w \in \mathbb{C} \ : \ |w-1| + |w+1| \leq 2\alpha\} \supseteq [-1, 1]$$

with $\alpha := g(\varrho)$, i.e., g is surjective from $\mathcal{A}_\varrho$ to $\mathcal{D}_\varrho$, cf. Figure 2.6.

Proof. Let $z \in \mathcal{A}_\varrho$ and $w := g(z)$. If $|z| = 1$, we have $g(z) \in [-1, 1] \subseteq \mathcal{D}_\varrho$.

If $|z| > 1$, we let $\hat{\varrho} := |z| \leq \varrho$, and Lemma 2.25 yields

$$|w-1| + |w+1| = 2g(\hat{\varrho}) \leq 2g(\varrho),$$

i.e., $w \in \mathcal{D}_\varrho$.

If $|z| < 1$ we have $w = g(z) = g(1/z)$ and can repeat the previous argument for $1/z$ instead of z due to $|1/z| = 1/|z| \leq \varrho$.

Let now $w \in \mathcal{D}_\varrho$. Let $\hat{\alpha} := \frac{1}{2}(|w-1| + |w+1|) \geq \frac{1}{2}(2 - |w+1| + |w+1|) = 1$. We also have $\hat{\alpha} \leq \alpha$ by definition and can use (2.30) to find $\hat{\varrho} \in [1, \varrho]$ with $g(\hat{\varrho}) = \hat{\alpha}$. Lemma 2.25 yields $w \in \mathcal{E}_{\hat{\varrho}} \subseteq \mathcal{D}_\varrho$. ∎

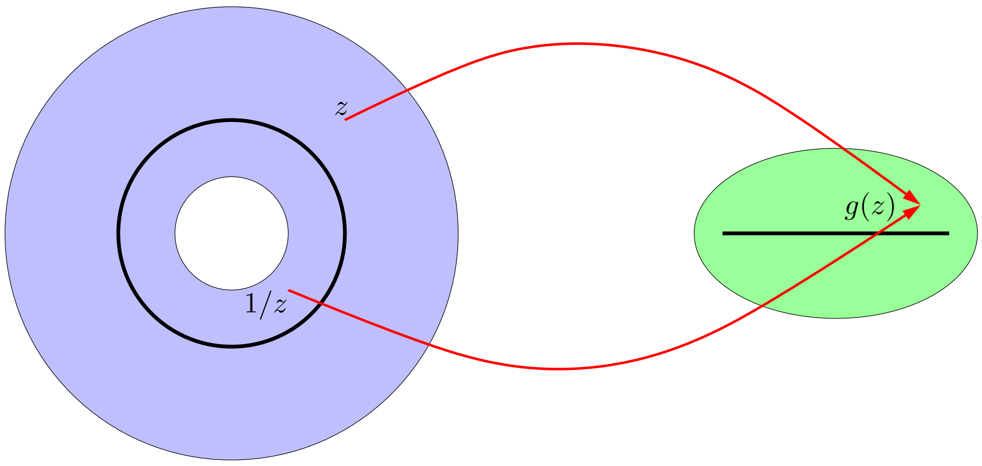

Figure 2.6: Joukowsky transformation from $\mathcal{A}_\varrho$ to $\mathcal{D}_\varrho$ with $\varrho = 2$

Let $\varrho \in \mathbb{R}_{>1}$, and let f be holomorphic in the Bernstein disc $\mathcal{D}_\varrho$. Then by Lemma 2.26, the function

$$\hat{f} := f \circ g$$

is holomorphic on the annulus $\mathcal{A}_\varrho$.

This means that $\hat{f}$ can be represented in a Laurent series, i.e., we have

$$\hat{f}(z) = \sum_{n=-\infty}^{\infty} a_n z^n \qquad \text{for all } z \in \mathcal{A}_\varrho \tag{2.33}$$

with the coefficients

$$a_n := \frac{1}{2\pi\iota} \int_{|z|=r} \frac{\hat{f}(z)}{z^{n+1}}\, dz \qquad \text{for all } n \in \mathbb{Z}, \tag{2.34}$$

for any $r \in (-1/\varrho, \varrho)$. We will now prove that the Laurent series (2.33) corresponds to the Chebyshev expansion (2.27).

Lemma 2.27 (Symmetry) *For all $n \in \mathbb{N}$, we have $a_n = a_{-n}$.*

Proof. Let $r \in (-1/\varrho, \varrho)$. We use the mappings

$$\gamma_1 \colon \mathbb{R} \to \mathbb{C}, \qquad t \mapsto re^{\iota t},$$
$$\gamma_2 \colon \mathbb{R} \to \mathbb{C}, \qquad 1 \mapsto \frac{1}{r} e^{\iota t} = \frac{1}{\gamma_1(-t)}$$

restricted to $[0, 2\pi]$ as parametrizations for the curve integrals over the circles with radii r and $1/r$ centered at zero. We have

$$\gamma_2'(t) = \frac{\gamma_1'(-t)}{\gamma_1(-t)^2}, \qquad \gamma_1'(t) = \gamma_1(t)^2 \gamma_2'(-t) = \frac{\gamma_2'(-t)}{\gamma_2(-t)^2} \qquad \text{for all } t \in [0, 2\pi].$$

Using this equation, $g(1/z) = g(z)$, and $\gamma_2(t - 2\pi) = \gamma_2(t)$, we find

$$\begin{aligned}
a_{-n} &= \frac{1}{2\pi\iota} \int_{|z|=r} \frac{\hat{f}(z)}{z^{-n+1}}\,dz = \frac{1}{2\pi\iota} \int_{|z|=r} \frac{\hat{f}(1/z)}{(1/z)^{n-1}}\,dz \\
&= \frac{1}{2\pi\iota} \int_0^{2\pi} \frac{\hat{f}(1/\gamma_1(t))}{(1/\gamma_1(t))^{n-1}} \gamma_1'(t)\,dt = \frac{1}{2\pi\iota} \int_0^{2\pi} \frac{\hat{f}(\gamma_2(-t))}{\gamma_2(-t)^{n-1}} \frac{\gamma_2'(-t)}{\gamma_2(-t)^2}\,dt \\
&= \frac{1}{2\pi\iota} \int_0^{2\pi} \frac{\hat{f}(\gamma_2(-t))}{\gamma_2(-t)^{n+1}} \gamma_2'(-t)\,dt = -\frac{1}{2\pi\iota} \int_0^{-2\pi} \frac{\hat{f}(\gamma_2(t))}{\gamma_2(t)^{n+1}} \gamma_2'(t)\,dt \\
&= \frac{1}{2\pi\iota} \int_{-2\pi}^{0} \frac{\hat{f}(\gamma_2(t))}{\gamma_2(t)^{n+1}} \gamma_2'(t)\,dt = \frac{1}{2\pi\iota} \int_0^{2\pi} \frac{\hat{f}(\gamma_2(t))}{\gamma_2(t)^{n+1}} \gamma_2'(t)\,dt \\
&= \frac{1}{2\pi\iota} \int_{|z|=1/r} \frac{\hat{f}(z)}{z^{n+1}}\,dz = a_n.
\end{aligned}$$

This is the required identity. ■

Using Lemma 2.27, the Laurent series (2.33) takes the form

$$\hat{f}(z) = a_0 + \sum_{n=1}^{\infty} a_n z^n + a_{-n} z^{-n} = a_0 + 2 \sum_{n=1}^{\infty} a_n \frac{z^n + z^{-n}}{2} \qquad \text{for all } z \in \mathcal{A}_\varrho. \tag{2.35}$$

To obtain the Chebyshev expansion (2.27), we have to prove that the functions $\frac{z^n+z^{-n}}{2}$ correspond to the Chebyshev polynomials (2.28).

Lemma 2.28 (Chebyshev polynomials) *We have*

$$C_n \circ g(z) = \frac{z^n + z^{-n}}{2} \qquad \textit{for all } n \in \mathbb{N}_0,\ z \in \mathbb{C} \setminus \{0\}. \tag{2.36}$$

Proof. By induction using the recurrence relation (2.28).

Let $z \in \mathbb{C} \setminus \{0\}$ and $w := g(z)$. For the base cases, we have

$$\begin{aligned}
C_0(w) &= 1 = \frac{2}{2} = \frac{z^0 + z^{-0}}{2}, \\
C_1(w) &= w = \frac{z + 1/z}{2} = \frac{z^1 + z^{-1}}{2}.
\end{aligned}$$

Now let $m \in \mathbb{N}$ be given such that (2.36) holds for all $n \in [0 : m]$. We have

$$\begin{aligned}
C_{m+1}(w) &= 2wC_m(w) - C_{m-1}(w) = 2\frac{z + 1/z}{2} \frac{z^m + z^{-m}}{2} - \frac{z^{m-1} + z^{1-m}}{2} \\
&= \frac{z^{m+1} + z^{m-1} + z^{1-m} + z^{-m-1}}{2} - \frac{z^{m-1} + z^{1-m}}{2} = \frac{z^{m+1} + z^{-(m+1)}}{2}.
\end{aligned}$$

This already completes the induction. ■

Using Lemma 2.28, the expansion (2.35) takes the form

$$f(g(z)) = a_0 + 2\sum_{n=1}^{\infty} a_n C_n(g(z)) \qquad \text{for all } z \in \mathcal{A}_\varrho,$$

and since the Joukowsky transformation g maps the annulus $\mathcal{A}_\varrho$ *surjectively* onto the Bernstein disc $\mathcal{D}_\varrho$, we conclude

$$f(w) = a_0 + 2\sum_{n=1}^{\infty} a_n C_n(w) \qquad \text{for all } w \in \mathcal{D}_\varrho. \tag{2.37}$$

This is called the *Chebyshev expansion* of the function f. In order to obtain a Chebyshev approximation, we "just" have to cut off the series after the first m terms.

Theorem 2.29 (Chebyshev approximation) *Let $m \in \mathbb{N}$, let $\hat\varrho \in [1, \varrho)$. The polynomial*

$$p := a_0 + 2\sum_{n=1}^{m} a_n C_n \in \Pi_m \tag{2.38}$$

satisfies

$$\|f - p\|_{\infty,\mathcal{D}_{\hat\varrho}} \le \frac{2}{\varrho/\hat\varrho - 1}\left(\frac{\hat\varrho}{\varrho}\right)^m \|f\|_{\infty,\mathcal{D}_\varrho}.$$

Proof. Let $w \in \mathcal{D}_{\hat\varrho}$. Due to Lemma 2.26, we can find $z \in \mathcal{A}_{\hat\varrho}$ with $g(z) = w$.

Let $n \in \mathbb{N}_0$. Due to Lemma 2.28, we have

$$|C_n(w)| = \frac{|z^n + z^{-n}|}{2} \le \frac{|z|^n + (1/|z|)^n}{2} \le \frac{\hat\varrho^n + \hat\varrho^n}{2} = \hat\varrho^n.$$

For all $r \in (1/\varrho, \varrho)$, the Laurent coefficient (2.34) can be bounded by

$$|a_n| \le \frac{1}{2\pi} 2\pi r \frac{\|\hat f\|_{\infty,\mathcal{A}_\varrho}}{r^{n+1}} = \frac{\|f\|_{\infty,\mathcal{D}_\varrho}}{r^n},$$

and we can go to the limit $r \to \varrho$ to obtain

$$|a_n| \le \frac{\|f\|_{\infty,\mathcal{D}_\varrho}}{\varrho^n}.$$

With these estimates at our disposal, we can directly prove the error estimate:

$$\begin{aligned}
|f(w) - p(w)| &= 2\left|\sum_{n=1}^{\infty} a_n C_n(w) - \sum_{n=1}^{m} a_n C_n(w)\right| = 2\left|\sum_{n=m+1}^{\infty} a_n C_n(w)\right| \\
&\le 2\sum_{n=m+1}^{\infty} |a_n|\,|C_n(w)| \le 2\|f\|_{\infty,\mathcal{D}_\varrho} \sum_{n=m+1}^{\infty} \frac{\hat\varrho^n}{\varrho^n} \\
&= 2\|f\|_{\infty,\mathcal{D}_\varrho} \left(\frac{\hat\varrho}{\varrho}\right)^{m+1} \sum_{n=0}^{\infty} \left(\frac{\hat\varrho}{\varrho}\right)^n = 2\|f\|_{\infty,\mathcal{D}_\varrho} \left(\frac{\hat\varrho}{\varrho}\right)^{m+1} \frac{1}{1 - \hat\varrho/\varrho}.
\end{aligned}$$

This is already the required result. ∎

Corollary 2.30 (Approximation on the interval) *Let $m \in \mathbb{N}$. The polynomial (2.38) satisfies*

$$\|f - p\|_{\infty,[-1,1]} \leq \frac{2}{\varrho - 1} \left(\frac{1}{\varrho}\right)^m \|f\|_{\infty,\mathcal{D}_\varrho}.$$

Proof. We apply Theorem 2.29 to $\hat{\varrho} = 1$ and $[-1,1] = \mathcal{D}_1$. ∎

This means that the rate of convergence is determined by the largest Bernstein disc in the domain of f.

If f is holomorphic up to a singularity at a point $w_0 \not\in [-1,1]$, we can find $\varrho_0 \in \mathbb{R}_{\geq 1}$ such that $w_0 \in \mathcal{E}_{\varrho_0}$. For every $\varrho \in (1, \varrho_0)$, the rate of convergence of the Chebyshev expansion (2.37) will be at least $1/\varrho$, i.e., asymptotically, we will have a rate of $1/\varrho_0$.

Example 2.31 (Analytic function) *We consider the function*

$$f\colon [-1,1] \to \mathbb{R}, \qquad\qquad x \mapsto \frac{1}{1+x^2}.$$

We cannot approximate it on the entire interval $[-1,1]$ by a Taylor expansion around the midpoint, since such an expansion would have to converge on the entire unit disc in the complex plane, and this disc would include the singularities at ι and $-\iota$.

For the Chebyshev expansion, the disc is replaced by a Bernstein ellipse. The Bernstein ellipse $\mathcal{E}_{\varrho_0}$ touching the singularities i and $-i$ is the one with

$$\begin{aligned} 2\sqrt{2} &= |i-1| + |i+1| = 2\alpha_0 = \varrho_0 + 1/\varrho_0, \\ 0 &= \varrho_0^2 - 2\sqrt{2}\varrho_0 + 1 = (\varrho_0 - \sqrt{2})^2 - 1, \end{aligned}$$

i.e., $\varrho_0 = \sqrt{2}+1$, and $1/\varrho_0 = \sqrt{2} - 1 \approx 0.41$ is indeed the rate of convergence observed in numerical experiments.

Example 2.32 (Logarithmic kernel) *In order to investigate the one-dimensional model problem, we apply $\Phi_{[a,b]}$ to $f(x) := g(x,y)$ with $y \in s$ and get*

$$\hat{f}(\hat{x}) = -\log|\Phi_{[a,b]}(\hat{x}) - y| = -\log\left|\frac{b+a}{2} - y + \frac{b-a}{2}\hat{x}\right|.$$

We introduce

$$w_0 := \frac{2}{b-a}\left(y - \frac{b+a}{2}\right) \in \mathbb{R}$$

and obtain

$$\hat{f}(\hat{x}) = -\log\left(\frac{b-a}{2}|w_0 - \hat{x}|\right),$$

i.e., the singularity is located at w_0. Let $x_0 \in t$ denote the point closest to y, i.e., $x_0 = a$ if $y < a$ and $x_0 = b$ if $y > b$. We have

$$|w_0| = \frac{2}{\operatorname{diam}(t)}\left|y - \frac{b+a}{2}\right| = \frac{2}{\operatorname{diam}(t)}\left|y - x_0 + x_0 - \frac{b+a}{2}\right|$$

$$\geq \frac{2(\operatorname{dist}(t,s) + \operatorname{diam}(t)/2)}{\operatorname{diam}(t)} = \frac{2\operatorname{dist}(t,s)}{\operatorname{diam}(t)} + 1.$$

Assuming that the admissibility condition (2.11) holds, we find

$$|w_0| \geq \frac{2\operatorname{dist}(t,s)}{\operatorname{diam}(t)} + 1 \geq \frac{1}{\eta} + 1 = \frac{\eta+1}{\eta}.$$

The Bernstein ellipse touching this point is the one with

$$2\frac{\eta+1}{\eta} = \frac{1}{\eta} + 2 + \frac{1}{\eta} = |w_0 - 1| + |w_0 + 1| = 2\alpha_0 = \varrho_0 + 1/\varrho_0,$$
$$0 = \varrho_0^2 - 2\frac{\eta+1}{\eta}\varrho_0 + 1 = \left(\varrho_0 - \frac{\eta+1}{\eta}\right)^2 - \frac{(\eta+1)^2}{\eta^2} + 1,$$

so we have

$$\varrho_0 = \frac{\eta + 1 + \sqrt{(\eta+1)^2 + \eta^2}}{\eta} > 2\frac{\eta+1}{\eta},$$

i.e., the rate of convergence is in fact more than twice that of the Taylor expansion. For $\eta = 1$*, we obtain* $\varrho_0 = 2 + \sqrt{5}$ *and* $1/\varrho_0 \approx 0.24$*, while* $\eta = 1/2$ *yields* $\varrho_0 = 3 + \sqrt{10}$ *and* $1/\varrho_0 \approx 0.16$*.*

According to Theorem 2.5, our choice $\eta = 1/2$ *yields only a rate of* $1/3 \approx 0.33$ *if we use the Taylor expansion.*

3 Multi-dimensional problems

The previous chapter has introduced the basic ideas of hierarchical matrix methods: the matrix G is subdivided into submatrices $G|_{\hat{t}\times\hat{s}}$, and these submatrices are approximated by low-rank factorizations $G|_{\hat{t}\times\hat{s}} \approx A_{ts}B_{ts}^*$.

Now our goal is to extend these concepts to significantly more general applications, i.e., more general domains and more general approximation schemes.

3.1 Gravitational potentials

Our model problem for this chapter is the evaluation of the classical gravitational potential resulting from suns distributed in d-dimensional space.

We describe the suns by a family $(m_j)_{j\in\mathcal{J}}$ of masses and a family $(y_j)_{j\in\mathcal{J}}$ of coordinates satisfying

$$m_j \in \mathbb{R}_{>0}, \qquad y_j \in \mathbb{R}^d \qquad \text{for all } j \in \mathcal{J},$$

where $\mathcal{J}$ is a suitable finite index set.

We want to evaluate the gravitational potential in a family $(x_i)_{i\in\mathcal{I}}$ of points satisfying

$$x_i \in \mathbb{R}^d \setminus \{y_j \ : \ j \in \mathcal{J}\} \qquad \text{for all } i \in \mathcal{I}.$$

The gravitational potential is given by

$$\phi_i := \sum_{j\in\mathcal{J}} \frac{\gamma}{\|x_i - y_j\|_2} m_j \qquad \text{for all } i \in \mathcal{I},$$

where $\gamma \in \mathbb{R}_{>0}$ denotes the gravitational constant. This computation can be interpreted as a matrix-vector multiplication: we define $G \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ by

$$g_{ij} := \frac{\gamma}{\|x_i - y_j\|_2} \qquad \text{for all } i \in \mathcal{I},\ j \in \mathcal{J} \tag{3.1}$$

and find

$$\phi = Gm.$$

Similar to our one-dimensional model problem, we have $g_{ij} > 0$ for all $i \in \mathcal{I}$ and $j \in \mathcal{J}$. With the function

$$g\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}, \qquad (x,y) \mapsto \begin{cases} \frac{\gamma}{\|x-y\|_2} & \text{if } x \neq y, \\ 0 & \text{otherwise,} \end{cases} \tag{3.2}$$

we have

$$g_{ij} = g(x_i, y_j) \qquad \text{for all } i \in \mathcal{I},\ j \in \mathcal{J},$$

so once again we have to deal with a kernel function that has a singularity for $x = y$.

3.2 Approximation by interpolation

In order to construct degenerate approximations for multi-dimensional kernel functions like the gravitational potential (3.2), we have to extend the results of section 2.6 from intervals to suitable subsets of of $\mathbb{R}^d$. This task is particularly straightforward if we consider axis-parallel boxes, i.e., domains

$$B = [a_1, b_1] \times \ldots \times [a_d, b_d], \qquad a_\iota < b_\iota \qquad \text{for all } \iota \in [1 : d]. \tag{3.3}$$

We recall that we can define transformed interpolation operators

$$\mathfrak{I}_{[a,b]} \colon C[a,b] \to \Pi_m, \qquad f \mapsto \mathfrak{I}[f \circ \Phi_{[a,b]}] \circ \Phi_{[a,b]}^{-1},$$

for arbitrary non-empty intervals $[a, b]$ and that these operators can be written in the form

$$\mathfrak{I}_{[a,b]}[f] = \sum_{\nu=0}^{m} f(\xi_{[a,b],\nu})\ell_{[a,b],\nu}$$

with transformed interpolation points $(\xi_{[a,b],\nu})_{\nu=0}^m$ and corresponding Lagrange polynomials $(\ell_{[a,b],\nu})_{\nu=0}^m$.

In order to extend the interpolation operator to the d-dimensional setting, we "freeze" all variables but one and apply interpolation to this one variable.

Definition 3.1 (Partial interpolation) *Let $\iota \in [1 : d]$, and let B denote the axis-parallel box given by (3.3). The operator $\mathfrak{I}_{B,\iota} : C(B) \to C(B)$ defined by*

$$\mathfrak{I}_{B,\iota}[f](x) = \sum_{\nu=0}^{m} f(x_1, \ldots, x_{\iota-1}, \xi_{[a_\iota,b_\iota],\nu}, x_{\iota+1}, \ldots, x_d)\, \ell_{[a_\iota,b_\iota],\nu}(x_\iota)$$
$$\text{for all } f \in C(B),\ x \in B$$

is called the ι-th partial interpolation operator *for B.*

The partial interpolation operator $\mathfrak{I}_{B,\iota}$ inherits many valuable properties of the one-dimensional interpolation operator $\mathfrak{I}_{[a_\iota,b_\iota]}$, but it does not produce a polynomial, merely a function that is polynomial with respect to the ι-th variable.

In order to obtain a "proper" interpolation operator, we have to apply partial interpolation operators to *all* coordinate directions $\iota \in [1 : d]$.

Definition 3.2 (Tensor interpolation) *The operator*

$$\mathfrak{I}_B := \mathfrak{I}_{B,1} \circ \cdots \circ \mathfrak{I}_{B,d}$$

is called the tensor interpolation operator *for B.*

We would like to obtain a representation similar to (2.15) for the tensor interpolation operator $\mathfrak{I}_B$, since this would in turn allow us to obtain approximations of the kernel function in the form (2.18).

To this end, we introduce the set $M := [0 : m]^d$ of *d-dimensional multi-indices* and let

$$\xi_{B,\nu} := (\xi_{[a_1,b_1],\nu_1}, \ldots, \xi_{[a_d,b_d],\nu_d}), \tag{3.4a}$$
$$\ell_{B,\nu}(x) := \ell_{[a_1,b_1],\nu_1}(x_1) \cdots \ell_{[a_d,b_d],\nu_d}(x_d) \qquad \text{for all } \nu \in M,\ x \in B. \tag{3.4b}$$

Using these *tensor interpolation points* and *tensor Lagrange polynomials*, the interpolation operator $\mathfrak{I}_B$ can be written in the desired form.

Lemma 3.3 (Lagrange representation) *We have*

$$\mathfrak{I}_B[f](x) = \sum_{\nu \in M} f(\xi_{B,\nu})\, \ell_{B,\nu}(x) \qquad \textit{for all } f \in C(B),\ x \in B. \tag{3.5}$$

Proof. In order to be able to apply induction, we introduce

$$\mathfrak{P}_0 := I, \qquad \mathfrak{P}_\iota := \mathfrak{P}_{\iota-1} \circ \mathfrak{I}_{B,\iota} \qquad \text{for all } \iota \in [1 : d]$$

and observe $\mathfrak{P}_d = \mathfrak{I}_B$. We let

$$\begin{aligned} M_\iota &:= [0 : m]^\iota, \\ \xi_{\iota,\nu} &:= (\xi_{[a_1,b_1],\nu_1}, \ldots, \xi_{[a_\iota,b_\iota],\nu_\iota}), \\ \ell_{\iota,\nu}(x) &:= \ell_{[a_1,b_1],\nu_1}(x_1) \cdots \ell_{[a_\iota,b_\iota],\nu_\iota}(x_\iota) \qquad \text{for all } \iota \in [1 : d],\ x \in \mathbb{R}^\iota \end{aligned}$$

and aim to prove

$$\mathfrak{P}_\iota[f](x) = \sum_{\nu \in M_\iota} f(\xi_{\iota,\nu}, x_{\iota+1}, \ldots, x_d)\, \ell_{\iota,\nu}(x_1, \ldots, x_\iota) \qquad \text{for all } f \in C(B),\ x \in B \tag{3.6}$$

by induction on $\iota \in [1 : d]$.

We start with $\iota = 1$. Let $f \in C(B)$ and $x \in B$. Due to $\mathfrak{P}_1 = \mathfrak{I}_{B,1}$, we can directly apply Definition 3.1 to obtain

$$\begin{aligned} \mathfrak{P}_1[f](x) = \mathfrak{I}_{B,1}[f](x) &= \sum_{\nu=0}^{m} f(\xi_{[a_1,b_1],\nu}, x_2, \ldots, x_d)\, \ell_{[a_1,b_1],\nu}(x_1) \\ &= \sum_{\nu \in M_1} f(\xi_{1,\nu}, x_2, \ldots, x_d)\, \ell_{1,\nu}(x_1). \end{aligned}$$

Let now $\iota \in [1 : d-1]$ be such that (3.6) holds. Let $f \in C(B)$ and $x \in B$. By the induction assumption, we have

$$\begin{aligned} \mathfrak{P}_{\iota+1}[f](x) &= \mathfrak{P}_\iota \circ \mathfrak{I}_{B,\iota+1}[f](x) \\ &= \mathfrak{P}_\iota[\mathfrak{I}_{B,\iota+1}[f]](x) = \sum_{\nu \in M_\iota} \mathfrak{I}_{B,\iota+1}[f](\xi_{\iota,\nu}, x_{\iota+1}, \ldots, x_d)\, \ell_{\iota,\nu}(x_1, \ldots, x_\iota) \end{aligned}$$

$$= \sum_{\nu\in M_\iota} \sum_{\mu=0}^{m} f(\xi_{\iota,\nu}, \xi_{[a_{\iota+1},b_{\iota+1}],\mu}, x_{\iota+2}, \ldots, x_d)\, \ell_{\iota,\nu}(x_1,\ldots,x_\iota)\, \ell_{[a_{\iota+1},b_{\iota+1}],\mu}(x_{\iota+1})$$
$$= \sum_{\hat\nu\in M_{\iota+1}} f(\xi_{\iota+1,\hat\nu}, x_{\iota+2},\ldots,x_d)\ell_{\iota+1,\hat\nu}(x_1,\ldots,x_{\iota+1}).$$

Applying (3.6) to $\iota = d$ and observing

$$M = M_d, \qquad \xi_{d,\nu} = \xi_{B,\nu}, \qquad \ell_{d,\nu} = \ell_{B,\nu} \qquad \text{for all } \nu\in M$$

completes the proof. ■

In order to approximate the kernel function g, we follow the approach outlined in the previous chapter, but replace the intervals t and s by axis-parallel boxes. The corresponding tensor interpolation operator

$$\mathfrak{I}_t[f] = \sum_{\nu\in M} f(\xi_{t,\nu})\, \ell_{t,\nu} \qquad \text{for all } f\in C(t)$$

can be used in the by now familiar way: we fix $y\in s$ and apply $\mathfrak{I}_t$ to the function $x\mapsto g(x,y)$, obtaining

$$\tilde g_{t,s}(x,y) := \sum_{\nu\in M} \ell_{t,\nu}(x)\, g(\xi_{t,\nu}, y) \qquad \text{for all } x\in t, y\in s, \tag{3.7}$$

and this is the multi-dimensional counterpart of (2.18).

Example 3.4 (Gravitational potential) *We can use (3.7) directly to obtain an approximation of the matrix G corresponding to the gravitational potential (3.1): assume that we have sets $\hat t \subseteq \mathcal{I}$ and $\hat s\subseteq \mathcal{J}$ such that $\tilde g_{t,s}$ is a good approximation of g for all target points x_i with $i\in\hat t$ and all source points y_j with $j\in\hat s$. We find*

$$g_{ij} = g(x_i,y_j) \approx \tilde g_{t,s}(x_i,y_j) = \sum_{\nu\in M} \ell_{t,\nu}(x_i)\, g(\xi_{t,\nu},y_j) = \sum_{\nu\in M} a_{ts,i\nu} b_{ts,j\nu} = (A_{ts}B_{ts}^*)_{ij} \tag{3.8}$$

with the matrices $A_{ts}\in\mathbb{R}^{\hat t\times M}$ and $B_{ts}\in\mathbb{R}^{\hat s\times M}$ given by

$$a_{ts,i\nu} := \ell_{t,\nu}(x_i), \qquad b_{ts,j\nu} := g(\xi_{t,\nu}, y_j) \qquad \text{for all } i\in\hat t,\ j\in\hat s,\ \nu\in M.$$

Preparing the entries of A_{ts} is fairly straightforward, we only have to evaluate the tensor Lagrange polynomials given by (3.4b). We can use the structure of these polynomials to compute rows of A_{ts} very efficiently.

Preparing the entries of B_{ts} is even more simple: we have to set up the tensor points (3.4a) and evaluate the kernel function (3.2).

In a practical implementation, handling the general index sets $\hat t$, $\hat s$, and M may pose a minor challenge. Usually $\hat t$ and $\hat s$ can be simply represented by arrays listing all indices, and these arrays give rise to a natural ordering of the rows of A_{ts} and B_{ts}. For the columns, the lexicographic *order on the multi-index set M is convenient: $\nu = (\nu_1,\nu_2,\nu_3)\in M$ is mapped to $\nu_1 + (m+1)(\nu_2 + (m+1)\nu_3) \in [0 : (m+1)^3 - 1]$.*

3.3 Error analysis

The degenerate kernel (3.7) is only useful if it provides us with a reasonable approximation of the original kernel function g, therefore we have to investigate the accuracy of tensor interpolation.

We start by taking a look at the partial interpolation operator and noticing that it inherits important properties of one-dimensional interpolation.

Lemma 3.5 (Partial interpolation) *Let $\iota \in [1:d]$, let B denote the axis-parallel box given by (3.3), let Λ_m be a stability constant satisfying (2.24), and let ω be the node polynomial defined by (2.20). We have*

$$\|\mathfrak{I}_{B,\iota}[f]\|_{\infty,B} \leq \Lambda_m \|f\|_{\infty,B} \qquad \text{for all } f \in C(B), \tag{3.9a}$$

$$\|f - \mathfrak{I}_{B,\iota}[f]\|_{\infty,B} \leq \|\omega\|_{\infty,[-1,1]} \left(\frac{b_\iota - a_\iota}{2}\right)^{m+1} \frac{\|\partial_\iota^{m+1} f\|_{\infty,B}}{(m+1)!} \qquad \text{for all } f \in C^{m+1}(B). \tag{3.9b}$$

Proof. Let $f \in C(B)$, and let $x \in B$. We define the function

$$\hat{f}\colon [a_\iota, b_\iota] \to \mathbb{R}, \qquad y \mapsto f(x_1, \ldots, x_{\iota-1}, y, x_{\iota+1}, \ldots, x_d)$$

and observe

$$\begin{aligned}
\mathfrak{I}_{B,\iota}[f](x) &= \sum_{\nu=0}^{m} f(x_1, \ldots, x_{\iota-1}, \xi_{[a_\iota,b_\iota],\nu}, x_{\iota+1}, \ldots, x_d)\, \ell_{[a_\iota,b_\iota],\nu}(x_\iota) \\
&= \sum_{\nu=0}^{m} \hat{f}(\xi_{[a_\iota,b_\iota],\nu})\, \ell_{[a_\iota,b_\iota],\nu}(x_\iota) = \mathfrak{I}_{[a_\iota,b_\iota]}[\hat{f}](x_\iota).
\end{aligned}$$

Due to (2.24), we have

$$\begin{aligned}
|\mathfrak{I}_{[a_\iota,b_\iota]}[\hat{f}](x_\iota)| &\leq \|\mathfrak{I}_{[a_\iota,b_\iota]}[\hat{f}]\|_{\infty,[a_\iota,b_\iota]} \leq \Lambda_m \|\hat{f}\|_{\infty,[a_\iota,b_\iota]} = \max\{|\hat{f}(y)| \ :\ y \in [a_\iota, b_\iota]\} \\
&= \max\{|f(x_1, \ldots, x_{\iota-1}, y, x_{\iota+1}, \ldots, x_d)| \ :\ y \in [a_\iota, b_\iota]\} \leq \|f\|_{\infty,B},
\end{aligned}$$

and this implies (3.9a). Let now $f \in C^{m+1}(B)$. This implies $\hat{f} \in C^{m+1}[a_\iota, b_\iota]$, and (2.22) yields

$$\begin{aligned}
|f(x) - \mathfrak{I}_{B,\iota}[f](x)| &= |\hat{f}(x_\iota) - \mathfrak{I}_{[a_\iota,b_\iota]}[\hat{f}](x_\iota)| \\
&\leq \|\omega\|_{\infty,[-1,1]} \left(\frac{b_\iota - a_\iota}{2}\right)^{m+1} \frac{\|\hat{f}^{(m+1)}\|_{\infty,[a_\iota,b_\iota]}}{(m+1)!} \\
&\leq \|\omega\|_{\infty,[-1,1]} \left(\frac{b_\iota - a_\iota}{2}\right)^{m+1} \frac{\|\partial_\iota^{m+1} f\|_{\infty,B}}{(m+1)!}.
\end{aligned}$$

This is already (3.9b). ■

Using the representation (3.5), the investigation of the multi-dimensional tensor interpolation can be reduced to the one-dimensional case.

Due to the special struction of the tensor interpolation operator, stability and approximation error estimates carry over directly from the partial interpolation operators.

Theorem 3.6 (Stability and approximation) *Let Λ_m denote the stability constant of (2.24). We have*

$$\begin{aligned} \|\mathfrak{I}_B[f]\|_{\infty,B} &\leq \Lambda_m^d \|f\|_{\infty,B}, \\ \|f - \mathfrak{I}_B[f]\|_{\infty,B} &\leq \sum_{\iota=1}^{d} \Lambda_m^{\iota-1} \|f - \mathfrak{I}_{B,\iota}[f]\|_{\infty,B} \qquad \textit{for all } f \in C(B). \end{aligned}$$

Proof. As in the proof of Lemma 3.3, we introduce

$$\mathfrak{P}_0 := I, \qquad \mathfrak{P}_\kappa := \mathfrak{P}_{\kappa-1} \circ \mathfrak{I}_{B,\kappa} \qquad \text{for all } \kappa \in [1:d].$$

We prove

$$\|\mathfrak{P}_\kappa[f]\|_{\infty,B} \leq \Lambda_m^\kappa, \tag{3.10a}$$

$$\|f - \mathfrak{P}_\kappa[f]\|_{\infty,B} \leq \sum_{\iota=1}^{\kappa} \Lambda_m^{\iota-1} \|f - \mathfrak{I}_{B,\iota}[f]\|_{\infty,B} \qquad \text{for all } f \in C(B) \tag{3.10b}$$

by induction on $\kappa \in [0:d]$.

For $\kappa = 0$, we have $\mathfrak{P}_\kappa = \mathfrak{P}_0 = I$ and (3.10) holds trivially.

Let now $\kappa \in [0:d-1]$ be such that (3.10) holds. Due to the induction assumption and Lemma 3.5, we have

$$\begin{aligned} \|\mathfrak{P}_{\kappa+1}[f]\|_{\infty,B} &= \|\mathfrak{P}_\kappa \circ \mathfrak{I}_{B,\kappa+1}[f]\|_{\infty,B} = \|\mathfrak{P}_\kappa[\mathfrak{I}_{B,\kappa+1}[f]]\|_{\infty,B} \\ &\leq \Lambda_m^\kappa \|\mathfrak{I}_{B,\kappa+1}[f]\|_{\infty,B} \leq \Lambda_m^\kappa \Lambda_m \|f\|_{\infty,B} = \Lambda_m^{\kappa+1} \|f\|_{\infty,B} \end{aligned}$$

for all $f \in C(B)$. This proves the stability estimate (3.10a).

We can use *both* induction assumptions and Lemma 3.5 to prove

$$\begin{aligned} \|f - \mathfrak{P}_{\kappa+1}[f]\|_{\infty,B} &= \|f - \mathfrak{P}_\kappa[f] + \mathfrak{P}_\kappa[f] - \mathfrak{P}_{\kappa+1}[f]\|_{\infty,B} \\ &\leq \|f - \mathfrak{P}_\kappa[f]\|_{\infty,B} + \|\mathfrak{P}_\kappa[f] - \mathfrak{P}_\kappa[\mathfrak{I}_{B,\kappa+1}[f]]\|_{\infty,B} \\ &= \|f - \mathfrak{P}_\kappa[f]\|_{\infty,B} + \|\mathfrak{P}_\kappa[f - \mathfrak{I}_{B,\kappa+1}[f]]\|_{\infty,B} \\ &\leq \|f - \mathfrak{P}_\kappa[f]\|_{\infty,B} + \Lambda_m^\kappa \|f - \mathfrak{I}_{B,\kappa+1}[f]\|_{\infty,B} \\ &\leq \sum_{\iota=1}^{\kappa} \Lambda_m^{\iota-1} \|f - \mathfrak{I}_{B,\iota}[f]\|_{\infty,B} + \Lambda_m^\kappa \|f - \mathfrak{I}_{B,\kappa+1}[f]\|_{\infty,B} \\ &= \sum_{\iota=1}^{\kappa+1} \Lambda_m^{\iota-1} \|f - \mathfrak{I}_{B,\iota}[f]\|_{\infty,B}. \end{aligned}$$

This proves the approximation error estimate (3.10b). ■

This is a particularly useful result, since it allows us to reduce the entire error analysis of tensor interpolation to the investigation of partial interpolation operators, which due to Lemma 3.5 behave essentially like one-dimensional interpolation operators.

Corollary 3.7 (Tensor interpolation error) *Let $\mathfrak{I}$ be the Chebyshev interpolation operator, and let Λ_m denote the stability constant of (2.24). We have*

$$\|f - \mathfrak{I}_B[f]\|_{\infty,B} \leq 2 \sum_{\iota=1}^{d} \Lambda_m^{\iota-1} \left(\frac{b_\iota - a_\iota}{4}\right)^{m+1} \frac{\|\partial_\iota^{m+1} f\|_{\infty,B}}{(m+1)!} \qquad \text{for all } f \in C^{m+1}(B).$$

Proof. We combine Theorem 3.6 with Lemma 3.5. Since we are using Chebyshev interpolation, we have $\|\omega\|_{\infty,[-1,1]} = 2^{-m}$. ∎

In order to turn this into a meaningful error estimate, we require bounds for the derivatives of $f(x) = g(x, y)$. This turns out to be surprisingly challenging, but complex analysis can help.

Lemma 3.8 (Holomorphic extension) *Let $x, p \in \mathbb{R}^n \setminus \{0\}$, and let $r := \|p\|_2$ and $\zeta := \|x\|_2/r$. There is a complex number $w \in \mathbb{C}$ such that $|w| = \|x\|_2$, and the function*

$$\hat{f} \colon \{z \in \mathbb{C} \;:\; |z| < \zeta\} \to \mathbb{C}, \qquad z \mapsto \frac{1}{\sqrt{(w + zr)(\bar{w} + zr)}},$$

is holomorphic and satisfies

$$\begin{aligned} |w + tr|^2 &= \|x + tp\|_2^2 && \text{for all } t \in \mathbb{R}, \\ \hat{f}(t) &= \frac{1}{\|x + tp\|_2} && \text{for all } t \in (-\zeta, \zeta), \\ |\hat{f}(z)| &\leq \frac{1}{\|x\|_2 - |z|\,\|p\|_2} && \text{for all } z \in \mathbb{C},\ |z| < \zeta. \end{aligned}$$

Proof. We can find an orthogonal transformation $Q \in \mathbb{R}^{n \times n}$ and $r \in \mathbb{R} \setminus \{0\}$ such that

$$\hat{p} := Qp = \begin{pmatrix} r \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

We introduce $\hat{x} := Qx$ and

$$w := \hat{x}_1 + \iota \left(\hat{x}_2^2 + \ldots + \hat{x}_n^2\right)^{1/2},$$

where $\iota \in \mathbb{C}$ denotes the complex unity. We observe $|w| = \|\hat{x}\|_2 = \|x\|_2$ and

$$\begin{aligned} |w + tr|^2 &= (w + tr)(\bar{w} + tr) = |w|^2 + tr(w + \bar{w}) + t^2 r^2 \\ &= \hat{x}_1^2 + \ldots + \hat{x}_n^2 + 2tr\hat{x}_1 + t^2 r^2 = (\hat{x}_1 + tr)^2 + \hat{x}_2^2 + \ldots + \hat{x}_n^2 \\ &= \|\hat{x} + t\hat{p}\|_2^2 = \|x + tp\|_2^2 \qquad \text{for all } t \in \mathbb{R}, \end{aligned}$$

i.e., we can switch from the n-dimensional space to the complex plane.

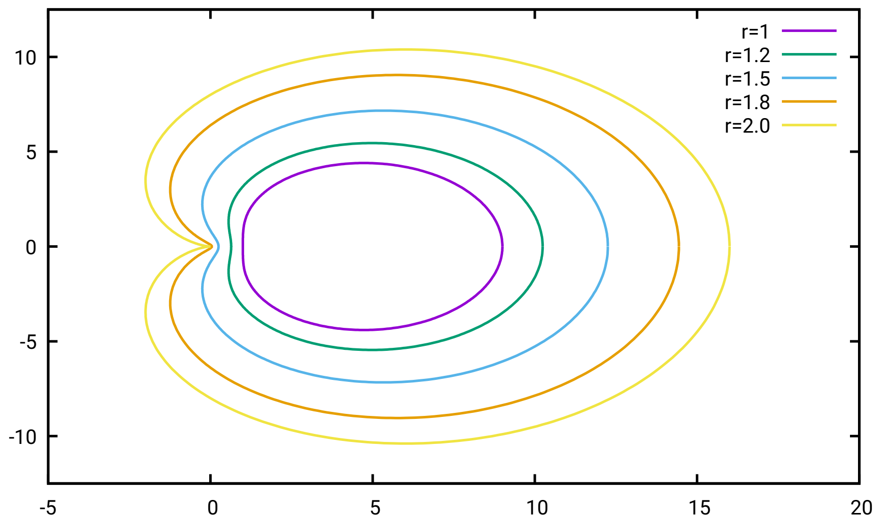

Figure 3.1: The set $\mathcal{S}$ for $|w| = 2$ and different values of r.

In order to define $\hat{f}$, we need a holomorphic extension of the square root to the set

$$\mathcal{S} := \{(w + zr)(\bar{w} + zr) \ : \ z \in \mathbb{C}, \ |z| < \zeta\}.$$

We would like to use the principal branch of the square root defined on $\mathbb{C} \setminus \mathbb{R}_{\leq 0}$ and therefore have to prove $\mathbb{R}_{\leq 0} \cap \mathcal{S} = \emptyset$.

Let $z = a + \iota b$ with $a, b \in \mathbb{R}$ and $|z| = \sqrt{a^2 + b^2} < \zeta$. We have

$$\begin{aligned}(w + zr)(\bar{w} + zr) &= |w|^2 + (w + \bar{w})zr + z^2r^2 = |w|^2 + 2\hat{x}_1 zr + z^2r^2 \\ &= |w|^2 + 2\hat{x}_1 ar + (a^2 - b^2)r^2 + \iota(2\hat{x}_1 br - 2abr^2).\end{aligned}$$

If the imaginary part is non-zero, this number is not in $\mathbb{R}_{\leq 0}$ and we are done.

If the imaginary part is zero, we have

$$2\hat{x}_1 br = 2abr^2 \iff b = 0 \vee \hat{x}_1 = ar.$$

If $b = 0$, we have

$$\begin{aligned}|w|^2 + 2\hat{x}_1 ar + (a^2 - b^2)r^2 &= |w|^2 + 2\hat{x}_1 ar + a^2r^2 \\ &\geq |w|^2 - 2|w|ar + a^2r^2 = (|w| - ar)^2 > 0,\end{aligned}$$

and due to $|ar| \leq |z|\, r < \zeta\, r = |w|$, the real part is strictly positive.

If $\hat{x}_1 = ar$, on the other hand, we have

$$\begin{aligned}|w|^2 + 2\hat{x}_1 ar + (a^2 - b^2)r^2 &= |w|^2 + 2a^2r^2 + a^2r^2 - b^2r^2 \geq |w|^2 + 3a^2r^2 - |z|^2r^2 \\ &> |w|^2 + 3a^2r^2 - \zeta^2r^2 = |w|^2 + 3a^2r^2 - |w|^2 = 3a^2r^2 \geq 0,\end{aligned}$$

i.e., the real part is again strictly positive.

We conclude that $\mathcal{S}$ is contained in the domain $\mathbb{C} \setminus \mathbb{R}_{\leq 0}$ of the principal branch of the square root and

$$\hat{f}(z) := \frac{1}{\sqrt{(w + zr)(\bar{w} + zr)}} \qquad \text{for all } z \in \mathbb{C},\ |z| < \zeta,$$

is a well-defined holomorphic function. We have

$$\begin{aligned}\hat{f}(t) &= \frac{1}{\sqrt{(w + tr)(\bar{w} + tr)}} = \frac{1}{\sqrt{|w + tr|^2}} \\ &= \frac{1}{\sqrt{\|\hat{x} + t\hat{p}\|^2}} = \frac{1}{\|\hat{x} + t\hat{p}\|} = \frac{1}{\|x + tp\|} \qquad \text{for all } t \in (-\zeta, \zeta),\end{aligned}$$

and

$$\begin{aligned}|\hat{f}(z)| &= \frac{1}{|\sqrt{(w + zr)(\bar{w} + zr)}|} = \frac{1}{\sqrt{|w + zr|}\,\sqrt{|\bar{w} + zr|}} \leq \frac{1}{\sqrt{|w| - |z|\,r}\sqrt{|\bar{w}| - |z|\,r}} \\ &= \frac{1}{|w| - |z|\,r} = \frac{1}{\|x\|_2 - |z|\,\|p\|_2} \qquad \text{for all } z \in \mathbb{C},\ |z| < \zeta.\end{aligned}$$

This completes the proof. ■

Lemma 3.9 (Derivatives) *Let $x, p \in \mathbb{R}^n \setminus \{0\}$. The ν-th directional derivative of $\hat{g}(x) := 1/\|x\|$ is bounded by*

$$\|\partial_p^\nu \hat{g}(x)\|_2 \leq (\nu + 1)!\, e \frac{\|p\|_2^\nu}{\|x\|_2^{\nu+1}} \qquad \textit{for all } \nu \in \mathbb{N}.$$

Proof. Let ζ, w and $\hat{f}$ be defined as in Lemma 3.8. We have

$$\partial_p^\nu \hat{g}(x) = \frac{\partial^\nu}{\partial t^\nu} \hat{g}(x + tp)\bigg|_{t=0} = \frac{\partial^\nu}{\partial t^\nu} \hat{f}(t)\bigg|_{t=0} = \hat{f}'(0)$$

and therefore $\partial_p^\nu \hat{g}(x) = \hat{f}^{(\nu)}(0)$. Since $\hat{f}$ is holomorphic in the circle of radius ζ around zero, we can use the Cauchy identity for derivatives.

Let $\epsilon \in (0, 1)$, set $\hat{\zeta} := (1 - \epsilon)\zeta$. We have

$$\begin{aligned}|\hat{f}^{(\nu)}(0)| &= \left| \frac{\nu!}{2\pi\iota} \int_{|z|=\hat{\zeta}} \frac{\hat{f}(z)}{z^{\nu+1}}\, dz \right| \leq \frac{\nu!}{2\pi} 2\pi\hat{\zeta} \frac{\max\{|\hat{f}(z)| \ :\ z \in \mathbb{C},\ |z| = \hat{\zeta}\}}{\hat{\zeta}^{\nu+1}} \\ &\leq \nu!\, \hat{\zeta}^{-\nu} \frac{1}{\|x\|_2 - \hat{\zeta}\|p\|_2} = \nu!\, \frac{\|p\|_2^\nu}{(1 - \epsilon)^\nu \|x\|_2^\nu} \frac{1}{\epsilon\|x\|_2} = \nu!\, \frac{\|p\|_2^\nu}{(1 - \epsilon)^\nu \epsilon \|x\|_2^{\nu+1}}.\end{aligned}$$

In order to reach our goal, we should choose an $\epsilon \in (0, 1)$ that makes $(1 - \epsilon)^\nu \epsilon$ as large as possible. We have

$$\frac{\partial}{\partial \epsilon}(1 - \epsilon)^\nu \epsilon = -\nu(1 - \epsilon)^{\nu-1}\epsilon + (1 - \epsilon)^\nu \qquad \text{for all } \epsilon \in (0, 1),$$

and computing the extremum at

$$\nu(1-\epsilon)^{\nu-1}\epsilon = (1-\epsilon)^\nu \iff \nu\epsilon = 1-\epsilon \iff 1 = (\nu+1)\epsilon \iff \epsilon = \frac{1}{\nu+1}$$

looks like a promising approach. Using this choice, we obtain

$$(1-\epsilon)^\nu = \left(1-\frac{1}{\nu+1}\right)^\nu = \left(\frac{\nu}{\nu+1}\right)^\nu = \left(\frac{\nu+1}{\nu}\right)^{-\nu} = \left(1+\frac{1}{\nu}\right)^{-\nu} > \frac{1}{e}$$

and

$$|\hat{f}^{(\nu)}(0)| < \nu!\, e\,(\nu+1)\frac{\|p\|_2^\nu}{\|x\|_2^{\nu+1}} = e\,(\nu+1)!\frac{\|p\|^\nu}{\|x\|^{\nu+1}}.$$

Due to $|\partial_p^\nu \hat{g}(x)| = |\hat{f}^{(\nu)}(0)|$, our proof is complete. ∎

We have

$$g(x,y) = \frac{\gamma}{\|x-y\|} = \gamma\,\hat{g}(x-y) \qquad \text{for all } x \in t,\ y \in s,$$

and in order to estimate the interpolation error

$$g(x,y) - \tilde{g}_{ts}(x,y) = f(x) - \mathfrak{I}_t[f](x) \qquad \text{for all } x \in t$$

with a fixed $y \in s$, we have to analyze the partial derivatives of

$$f(x) = g(x,y) = \gamma\,\hat{g}(x-y) \qquad \text{for all } x \in t.$$

According to Lemma 3.9, we have to be able to bound both $\|x-y\|$ from below and $|b_\iota - a_\iota|$ from above. To obtain a similar result as in the previous chapter, we introduce

$$\operatorname{diam}(t) := \max\{|b_\iota - a_\iota| \ :\ \iota \in [1:d]\},$$
$$\operatorname{dist}(t,s) := \inf\{\|x-y\| \ :\ x \in t,\ y \in s\}$$

and apply the previous Lemma.

Corollary 3.10 (Kernel function) *Let* $\operatorname{dist}(t,s) > 0$. *We have*

$$\|g - \tilde{g}_{ts}\|_{\infty,t\times s} \le \frac{2e\gamma d(m+2)\Lambda_m^{d-1}}{\operatorname{dist}(t,s)} \left(\frac{\operatorname{diam}(t)}{4\operatorname{dist}(t,s)}\right)^{m+1} \qquad \textit{for all } m \in \mathbb{N}.$$

Proof. Let $y \in s$. We define again $f(x) := g(x,y) = \gamma\,\hat{g}(x-y)$ for all $x \in t$. Due to Lemma 3.9, we have

$$\|\partial_\iota^{m+1} f\|_{\infty,t} \le \gamma(m+2)!\, e\frac{1}{\operatorname{dist}(t,s)^{m+2}} \qquad \text{for all } \iota \in [1:d],$$

and Corollary 3.7 yields

$$\|f - \mathfrak{I}[f]\|_{\infty,t} \le 2\sum_{\iota=1}^{d} \Lambda_m^{\iota-1}\left(\frac{b_\iota - a_\iota}{4}\right)^{m+1} \gamma\frac{(m+2)!\,e}{(m+1)!}\frac{1}{\operatorname{dist}(t,s)^{m+2}}$$
$$\le \frac{2ed\Lambda_m^{d-1}\gamma(m+2)}{\operatorname{dist}(t,s)}\left(\frac{\operatorname{diam}(t)}{4\ \operatorname{dist}(t,s)}\right)^{m+1}.$$

This is already the required estimate for the interpolation error. ∎

3.4 Cluster tree and block tree

Corollary 3.10 states that we can expect fast convergence of the kernel approximation if

$$\frac{\operatorname{diam}(t)}{4\operatorname{dist}(t,s)}$$

is small, at least less than one. As in the one-dimensional model problem, we use an *admissibility condition*

$$\operatorname{diam}(t) \leq 2\eta \operatorname{dist}(t,s) \tag{3.11}$$

with a parameter $\eta \in (0,2)$ that can be used to balance the rate of convergence versus the storage requirements. If (3.11) holds, Corollary 3.10 leads us to expect a rate of approximately $\eta/2$, since Λ_m^{d-1} and $m+2$ grow only slowly while $(\eta/2)^{m+1}$ decays exponentially.

To ensure (3.11), we proceed as in the previous chapter, i.e., we split the domains t and s into subdomains until they are either admissible or small enough to be handled directly. In our example, the boxes t and s correspond to clusters of suns, therefore we will refer to them as *clusters* in the following.

In order to split a cluster t, we generalize the bisection approach introduced in section 2.4. Assume that

$$t = [a_1,b_1] \times \cdots \times [a_d,b_d]$$

is too large. We choose a coordinate index $\iota \in [1:d]$ and define

$$c_\iota := \frac{b_\iota + a_\iota}{2}.$$

We split t into two halves along the plane $x_\iota = c_\iota$, i.e., we define

$$\begin{aligned} t_1 &:= \{x \in t \ : \ x_\iota \leq c_\iota\} \\ &= [a_1,b_1] \times \cdots \times [a_{\iota-1},b_{\iota-1}] \times [a_\iota,c_\iota] \times [a_{\iota+1},b_{\iota+1}] \times \cdots \times [a_d,b_d], \\ t_2 &:= \{x \in t \ : \ x_\iota \geq c_\iota\} \\ &= [a_1,b_1] \times \cdots \times [a_{\iota-1},b_{\iota-1}] \times [c_\iota,b_\iota] \times [a_{\iota+1},b_{\iota+1}] \times \cdots \times [a_d,b_d]. \end{aligned}$$

If t_1 and t_2 are still too large, we can repeat the procedure recursively.

Remark 3.11 (Coordinate selection) *A simple and effective strategy for choosing the coordinate index $\iota \in [1:d]$ for splitting a cluster is to go by the maximal extent, i.e., to choose $\iota \in [1:d]$ such that*

$$b_\iota - a_\iota \geq b_\kappa - a_\kappa \qquad \text{for all } \kappa \in [1:d],$$

since this guarantees that diam(t) *shrinks as rapidly as possible. We will refer to this approach as* adaptive coordinate selection.

For theoretical investigations, the regular coordinate selection *approach is sometimes more attractive: we use $\iota = 1$ for the first cluster, $\iota = 2$ for the second generation,*

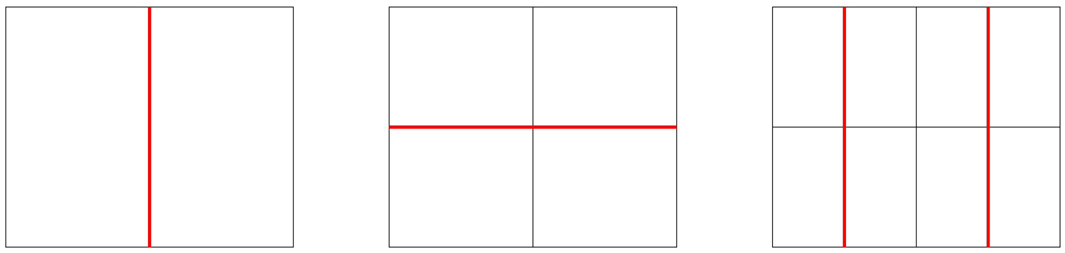

Figure 3.2: Cluster construction by bisection

$\iota = d$ for the d-th generation, and then we start again with $\iota = 1$ for the $(d+1)$-th generation. This strategy ensures that the ratios of the sides of the clusters obtained in the d-th generation are identical to those we started with, a useful property for the geometric complexity analysis.

Every cluster t is associated with a set $\hat{t} \subseteq \mathcal{I}$ of indices such that $x_i \in t$ for all $i \in \hat{t}$, and it is frequently a good idea to construct these sets together with the corresponding clusters. In our case, the sets

$$\hat{t}_1 := \{i \in \hat{t} \ : \ x_{i,\iota} \leq c_\iota\},$$
$$\hat{t}_2 := \{i \in \hat{t} \ : \ x_{i,\iota} > c_\iota\}$$

are an obvious choice. We want $\hat{t}_1$ and $\hat{t}_2$ to be disjoint, since every point should be approximated only once, and we ensure this by including $i \in \hat{t}$ with $x_{i,\iota} = c_\iota$ only in the first domain.

Remark 3.12 (A posteriori index sets) *Computing the index sets $\hat{t}$ during the construction of the clusters is attractive since it allows us to stop subdividing as soon as a cluster contains only a small number of indices.*

The price for this advantage is that sorting indices into the clusters typically takes $\mathcal{O}(n)$ operations for each generation, where $n = |\hat{t}|$ is the cardinality of the index set in the first generation. Since we usually need $\mathcal{O}(\log n)$ generations to arrive at sufficiently small subdomains, this approach can be expected to take at least $\mathcal{O}(n \log n)$ operations.

An alternative is to construct all generations without considering the index sets. Once the final generation has been created, the indices are distributed to the last-generation clusters. If we store the indices in nested arrays, i.e., if the arrays corresponding to child clusters are sub-arrays of the array corresponding to the parent, filling the last generation index sets also fills all other index sets, and we obtain linear complexity $\mathcal{O}(n)$.

We can see that the bisection procedure gives rise to a hierarchical decomposition of clusters and index sets: the clusters can be arranged in a tree, using the original domains as the root and creating children of a cluster by bisection.

Definition 3.13 (Tree) *Let V be a finite set, let $r \in V$ and $E \subseteq V \times V$. We call $\mathcal{T} := (V, r, E)$ a* tree *if for each $v \in V$ there is exactly one sequence $v_0, v_1, \ldots, v_\ell \in V$, $\ell \in \mathbb{N}_0$, such that*

$$v_0 = r, \qquad v_\ell = v, \qquad (v_{i-1}, v_i) \in E \qquad \textit{for all } i \in [1 : \ell].$$

We call V the nodes *of the tree, r its* root*, and E its* edges*. The root is denoted by* $\operatorname{root}(\mathcal{T}) = r$*. We frequently use $v \in \mathcal{T}$ as short-hand for $v \in V$.*

For each $v \in \mathcal{T}$, we call

$$\operatorname{chil}(v) := \{w \in V \ : \ (v, w) \in E\}$$

the set of children *of v. If* $\operatorname{chil}(v) = \emptyset$*, we call v a* leaf*.*

If there is a $w \in V$ such that $(w, v) \in E$, this w is unique and is called the parent *of v. Our definition implies that only the root r has no parent.*

Definition 3.14 (Cluster tree) *Let $\mathcal{I}$ be a finite set, let $\mathcal{T}_{\mathcal{I}}$ be a tree with nodes $t \in \mathcal{T}_{\mathcal{I}}$, and let a subset $\hat{t} \subseteq \mathcal{I}$ be given for each $t \in \mathcal{T}_{\mathcal{I}}$.*

We call it a cluster tree *for $\mathcal{I}$ if*

- *the index set corresponding to the root r is $\mathcal{I}$, i.e.,*

$$\hat{r} = \mathcal{I}, \tag{3.12a}$$

- *the index set corresponding to a non-leaf node consists of the union of the index sets of its children, i.e.,*

$$\hat{t} = \bigcup_{t' \in \operatorname{chil}(t)} \hat{t}' \qquad \textit{for all } t \in \mathcal{T}_{\mathcal{I}},\ \operatorname{chil}(t) \neq \emptyset,\ \textit{and} \tag{3.12b}$$

- *the index sets of different children of a node are disjoint, i.e.,*

$$\hat{t}_1 \cap \hat{t}_2 \neq \emptyset \Rightarrow t_1 = t_2 \qquad \textit{for all } t \in \mathcal{T}_{\mathcal{I}},\ t_1, t_2 \in \operatorname{chil}(t). \tag{3.12c}$$

The nodes $t \in \mathcal{T}_{\mathcal{I}}$ of a cluster tree $\mathcal{T}_{\mathcal{I}}$ are called clusters*. The set of leaves of $\mathcal{T}_{\mathcal{I}}$ is denoted by $\mathcal{L}_{\mathcal{I}} := \{t \in \mathcal{T}_{\mathcal{I}} \ : \ \operatorname{chil}(t) = \emptyset\}$.*

Splitting the clusters into *levels* is frequently a very useful tool, both for proving statements about trees and sometimes also for implementing tree algorithms.

Definition 3.15 (Level) *Let $\mathcal{T}_{\mathcal{I}}$ be a cluster tree with root $r := \operatorname{root}(\mathcal{T}_{\mathcal{I}})$. The* level *of a cluster $t \in \mathcal{T}_{\mathcal{I}}$ is defined by*

$$\operatorname{level}(t) := \begin{cases} 0 & \textit{if } t = r, \\ \operatorname{level}(t_+) + 1 & \textit{if there exists } t_+ \in \mathcal{T}_{\mathcal{I}} \textit{ with } t \in \operatorname{chil}(t_+) \end{cases} \qquad \textit{for all } t \in \mathcal{T}_{\mathcal{I}}.$$

The maximal level

$$p_{\mathcal{I}} := \max\{\operatorname{level}(t) \ : \ t \in \mathcal{T}_{\mathcal{I}}\}$$

is called the depth *of the cluster tree.*

Lemma 3.16 (Leaf indices) *Let $\mathcal{T}_{\mathcal{I}}$ be a cluster tree for the index set $\mathcal{I}$. For every $i \in \mathcal{I}$, there exists a leaf $t \in \mathcal{L}_{\mathcal{I}}$ with $i \in \hat{t}$.*

Proof. Let $i \in \mathcal{T}_{\mathcal{I}}$. The set

$$C_i := \{t \in \mathcal{T}_{\mathcal{I}} \ : \ i \in \hat{t}\}$$

contains the root due to (3.12a), so it is not empty.

If a cluster $t \in C_i$ has children, (3.12b) yields that there has to be $t' \in \operatorname{chil}(t)$ with $i \in \hat{t}'$ and therefore $t' \in C_i$. Definition 3.15 yields $\operatorname{level}(t') = \operatorname{level}(t) + 1$.

Since C_i is finite and not empty, we can choose a $t \in C_i$ such that $\operatorname{level}(t)$ is maximal. We have seen that this implies $\operatorname{chil}(t) = \emptyset$, i.e., we have $t \in \mathcal{L}_{\mathcal{I}}$ and $i \in \hat{t}$. ■

Definition 3.17 (Descendants and predecessors) *Let $\mathcal{T}_{\mathcal{I}}$ be a cluster tree. We define the sets of* descendants

$$\operatorname{desc}(t) := \{t\} \cup \bigcup_{t' \in \operatorname{chil}(t)} \operatorname{desc}(t') \qquad \textit{for all } t \in \mathcal{T}_{\mathcal{I}}$$

and predecessors

$$\operatorname{pred}(t) := \{t_* \in \mathcal{T}_{\mathcal{I}} \ : \ t \in \operatorname{desc}(t_*)\} \qquad \textit{for all } t \in \mathcal{T}_{\mathcal{I}}.$$

Lemma 3.18 (Intersecting clusters) *Let $t, s \in \mathcal{T}_{\mathcal{I}}$ with $\hat{t} \cap \hat{s} \neq \emptyset$.*

If $\operatorname{level}(s) = \operatorname{level}(t)$, *we have* $t = s$. *If* $\operatorname{level}(s) \leq \operatorname{level}(t)$, *we have* $t \in \operatorname{desc}(s)$.

Proof. We first prove that $\operatorname{level}(t) = \operatorname{level}(s)$ and $\hat{t} \cap \hat{s} \neq \emptyset$ implies $t = s$ by induction on $\operatorname{level}(t) \in \mathbb{N}_0$.

Let $t, s \in \mathcal{T}_{\mathcal{I}}$ with $\operatorname{level}(t) = \operatorname{level}(s) = 0$. Then both clusters are the root and therefore identical.

Let now $\ell \in \mathbb{N}_0$ be given such that for all $t, s \in \mathcal{T}_{\mathcal{I}}$ the identity $\operatorname{level}(t) = \operatorname{level}(s) = \ell$ and $\hat{t} \cap \hat{s} \neq \emptyset$ imply $t = s$.

Let $t, s \in \mathcal{T}_{\mathcal{I}}$ with $\operatorname{level}(t) = \operatorname{level}(s) = \ell + 1$ and $\hat{t} \cap \hat{s} \neq \emptyset$. Since $\operatorname{level}(t), \operatorname{level}(s) > 0$, there are parent clusters $t_+, s_+ \in \mathcal{T}_{\mathcal{I}}$ with $t \in \operatorname{chil}(t_+)$ and $s \in \operatorname{chil}(s_+)$ and $\operatorname{level}(t_+) = \operatorname{level}(s_+) = \ell$. Due to (3.12b), we have $\hat{t}_+ \cap \hat{s}_+ \supseteq \hat{t} \cap \hat{s} \neq \emptyset$, so we can apply the induction assumption to obtain $t_+ = s_+$. This means that $t, s \in \operatorname{chil}(t_+)$, and (3.12c) yields $t = s$.

We now prove that $\operatorname{level}(t) - \operatorname{level}(s) \in \mathbb{N}_0$ and $\hat{t} \cap \hat{s} \neq \emptyset$ implies $t \in \operatorname{desc}(s)$ by induction on $\operatorname{level}(t) - \operatorname{level}(s) \in \mathbb{N}_0$.

Let $t, s \in \mathcal{T}_{\mathcal{I}}$ with $\operatorname{level}(t) - \operatorname{level}(s) = 0$ and $\hat{t} \cap \hat{s} \neq \emptyset$. By the first part of this proof, this implies $t = s$.

Let now $m \in \mathbb{N}_0$ be given such that for all $t, s \in \mathcal{T}_{\mathcal{I}}$ the identity $\operatorname{level}(t) - \operatorname{level}(s) = m$ and $\hat{t} \cap \hat{s} \neq \emptyset$ imply $t \in \operatorname{desc}(s)$.

Let $t, s \in \mathcal{T}_{\mathcal{I}}$ with $\operatorname{level}(t) - \operatorname{level}(s) = m+1$ and $\hat{t} \cap \hat{s} \neq \emptyset$. Since $\operatorname{level}(t) > \operatorname{level}(s) \geq 0$, there is a parent cluster $t_+ \in \mathcal{T}_{\mathcal{I}}$ with $t \in \operatorname{chil}(t_+)$. We have $\operatorname{level}(t_+) - \operatorname{level}(s) = \operatorname{level}(t) - 1 - \operatorname{level}(s) = m$ and $\hat{t}_+ \cap \hat{s} \supseteq \hat{t} \cap \hat{s} \neq \emptyset$, so we can apply the induction assumption to obtain $t_+ \in \operatorname{desc}(s)$. $t \in \operatorname{desc}(s)$ follows by definition. ■

Corollary 3.19 (Leaf partition) *The set $\{\hat{t} \;:\; t \in \mathcal{L}_{\mathcal{I}}\}$ is a disjoint partition of $\mathcal{I}$.*

Proof. Lemma 3.16 provides that every $i \in \mathcal{I}$ appears in a leaf cluster.

Lemma 3.18 provides that for all $t, s \in \mathcal{L}_{\mathcal{I}}$ with $\hat{t} \cap \hat{s} \neq \emptyset$, we have either $t \in \operatorname{desc}(s)$ or $s \in \operatorname{desc}(t)$. Since both are leaves, this is only possible if $t = s$. ■

Once we have cluster trees $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$ for target and source points at our disposal, we can look for admissible pairs of clusters. We can use a recursive procedure: given clusters t and s, we check the admissibility condition (3.11). If it holds, we stop and apply our low-rank approximation. If it does not hold, we check whether t and s have children. If they do, we recursively check pairs of children. If both are leaves, we stop and use a standard representation. This approach again gives rise to a tree structure.

Definition 3.20 (Block tree) *Let $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$ be cluster trees for index sets $\mathcal{I}$ and $\mathcal{J}$. A tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is called a* block tree *for $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$ if*

- *the nodes are pairs of clusters, i.e., for every $b \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ there are $t \in \mathcal{T}_{\mathcal{I}}$ and $s \in \mathcal{T}_{\mathcal{J}}$ with $b = (t, s)$,*
- *the root consists of the roots of $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$, i.e.,*

$$r = (\operatorname{root}(\mathcal{T}_{\mathcal{I}}), \operatorname{root}(\mathcal{T}_{\mathcal{J}})), \quad \textit{and} \tag{3.13a}$$

- *if $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ has children, they are pairs of the children of t and s, i.e.,*

$$\operatorname{chil}(b) = \begin{cases} \operatorname{chil}(t) \times \operatorname{chil}(s) & \textit{if } \operatorname{chil}(t) \neq \emptyset, \operatorname{chil}(s) \neq \emptyset, \\ \{t\} \times \operatorname{chil}(s) & \textit{if } \operatorname{chil}(t) = \emptyset, \operatorname{chil}(s) \neq \emptyset, \\ \operatorname{chil}(t) \times \{s\} & \textit{if } \operatorname{chil}(t) \neq \emptyset, \operatorname{chil}(s) = \emptyset. \end{cases} \tag{3.13b}$$

The nodes of a block tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ are called blocks. *For a block $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, $t \in \mathcal{T}_{\mathcal{I}}$ is called the* row cluster *or* target cluster *and $s \in \mathcal{T}_{\mathcal{J}}$ is called the* column cluster *or* source cluster.

The set of leaves of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is denoted by $\mathcal{L}_{\mathcal{I}\times\mathcal{J}}$.

For every block $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, we let $\hat{b} := \hat{t} \times \hat{s} \subseteq \mathcal{I} \times \mathcal{J}$.

We introduce an abbreviation to avoid the special cases when treating (3.13b).

Lemma 3.21 (Extended children) *We let*

$$\operatorname{chil}^+(t) := \begin{cases} \operatorname{chil}(t) & \textit{if } \operatorname{sons}(t) \neq \emptyset, \\ \{t\} & \textit{otherwise} \end{cases} \qquad \textit{for all } t \in \mathcal{T}_{\mathcal{I}}. \tag{3.14}$$

We have

$$\hat{t} = \bigcup_{t' \in \operatorname{chil}^+(t)} \hat{t}' \qquad \textit{for all } t \in \mathcal{T}_{\mathcal{I}},$$

$$\hat{t}_1 \cap \hat{t}_2 \neq \emptyset \Rightarrow t_1 = t_2 \qquad \textit{for all } t \in \mathcal{T}_{\mathcal{I}},\ t_1, t_2 \in \operatorname{chil}^+(t).$$

Proof. Follows directly from Definition 3.14. ■

As a first application of this notation, we prove that block trees are, in fact, cluster trees for the product index set $\mathcal{I} \times \mathcal{J}$, i.e., the statements we have proven so far for cluster trees carry over directly to block trees.

Lemma 3.22 (Product cluster tree) *Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a block tree for $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$. Then $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is a cluster tree for the product index set $\mathcal{I} \times \mathcal{J}$.*

Proof. Let $r = \operatorname{root}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}})$. By definition, we have $r = (t, s)$ with $t = \operatorname{root}(\mathcal{T}_{\mathcal{I}})$ and $s = \operatorname{root}(\mathcal{T}_{\mathcal{J}})$, and (3.12a) implies $\hat{t} = \mathcal{I}$ and $\hat{s} = \mathcal{J}$. Due to $\hat{r} = \hat{t} \times \hat{s} = \mathcal{I} \times \mathcal{J}$, we have proven (3.12a) for $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$.

Let now $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be given with $\operatorname{chil}(b) \neq \emptyset$. (3.13b) yields

$$\operatorname{chil}(b) = \operatorname{chil}^+(t) \times \operatorname{chil}^+(s),$$

and we can use Lemma 3.21 to obtain

$$\bigcup_{b' \in \operatorname{chil}(b)} \hat{b}' = \bigcup_{t' \in \operatorname{chil}^+(t)} \bigcup_{s' \in \operatorname{chil}^+(s)} \hat{t}' \times \hat{s}' = \left(\bigcup_{t' \in \operatorname{chil}^+(t)} \hat{t}' \right) \times \left(\bigcup_{s' \in \operatorname{chil}^+(s)} \hat{s}' \right) = \hat{t} \times \hat{s} = \hat{b}.$$

This proves (3.12b).

Let now $b_1, b_2 \in \operatorname{chil}(b)$ with $\hat{b}_1 \cap \hat{b}_2 \neq \emptyset$. By Definition 3.20, we can find row clusters $t_1, t_2 \in \operatorname{chil}^+(t)$ and column clusters $s_1, s_2 \in \operatorname{chil}^+(s)$ such that $b_1 = (t_1, s_1)$ and $b_2 = (t_2, s_2)$. Since we have

$$(\hat{t}_1 \cap \hat{t}_2) \times (\hat{s}_1 \cap \hat{s}_2) = (\hat{t}_1 \times \hat{s}_1) \cap (\hat{t}_2 \times \hat{s}_2) = \hat{b}_1 \cap \hat{b}_2 \neq \emptyset,$$

we find $\hat{t}_1 \cap \hat{t}_2 \neq \emptyset$ and $\hat{s}_1 \cap \hat{s}_2 \neq \emptyset$. Lemma 3.21 yields $t_1 = t_2$ and $s_1 = s_2$, and this implies $b_1 = b_2$. ■

Corollary 3.23 (Block partition) *Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a block tree for $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$. Then*

$$\{\hat{t} \times \hat{s} \ : \ b = (t, s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}\}$$

is a disjoint partition of $\mathcal{I} \times \mathcal{J}$.

Proof. Combine Lemma 3.22 with Corollary 3.19. ■

This result is at the heart of our approximation strategy: it allows us to construct an approximation of a matrix $G \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ by combining approximations of submatrices $G|_{\hat{t}\times\hat{s}}$ for $b = (t, s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$.

3.5 Hierarchical matrices

Our construction of a block tree stops if one of two conditions is met: either a block $b = (t, s)$ satisfies an admissibility condition like (2.11) or (3.11), or if a block cannot be subdivided any further.

In the first case, we use a low-rank approximation $G|_{\hat{t}\times\hat{s}} \approx A_{ts}B_{ts}^*$, while in the second case we assume that the matrix $G|_{\hat{t}\times\hat{s}}$ is sufficiently small to allow us to store it directly. This difference between leaf blocks gives rise to the following definition:

Definition 3.24 (Admissible and inadmissible leaves) *Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a block tree for cluster trees $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$. We split the set $\mathcal{L}_{\mathcal{I}\times\mathcal{J}}$ of the leaves of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ into*

- *a set of* admissible *leaves $\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}} \subseteq \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$ corresponding to the given admissibility condition, and*
- *a set of* inadmissible leaves $\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}} := \mathcal{L}_{\mathcal{I}\times\mathcal{J}} \setminus \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$.

Due to Corollary 3.23, the leaves of the block tree give rise to a partition of the index set $\mathcal{I} \times \mathcal{J}$, and therefore to a decomposition of a matrix $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ into submatrices.

Notation 3.25 (Matrix notations) *Since most of our algorithms work for both real- and complex-valued matrices, we introduce $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$ as a placeholder for the field currently under consideration.*

If $\mathcal{I}$ is a general index set and $k \in \mathbb{N}$, we use the abbreviation $\mathbb{K}^{\mathcal{I}\times k} := \mathbb{K}^{\mathcal{I}\times[1:k]}$.

For a general matrix $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$, X^* denotes the adjoint with respect to the Euclidean inner product, i.e., $X^* = X^T$ if $\mathbb{K} = \mathbb{R}$ and $X^* = X^H$ if $\mathbb{K} = \mathbb{C}$.

In practice, we frequently cannot work with the matrices $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ corresponding to our model problems, since they are simple too large. That is why we replace them with blockwise low-rank approximations, and these approximations have a structure that is worth a far closer look, since it can be used in far more general applications than the ones we have seen so far.

Definition 3.26 (Hierarchical matrix) *Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a block tree for cluster trees $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$, and let $\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ denote its admissible leaves. Let $k \in \mathbb{N}_0$.*

A matrix $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ is called a hierarchical matrix *(or $\mathcal{H}$-*matrix*) of local rank k for $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ if for every admissible leaf $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, there are matrices $A_b \in \mathbb{K}^{\hat{t}\times k}$ and $B_b \in \mathbb{K}^{\hat{s}\times k}$ such that*

$$G|_{\hat{t}\times\hat{s}} = A_b B_b^*. \tag{3.15}$$

The set of all such hierarchical matrices is denoted by $\mathcal{H}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, k)$.

The mere existence of the factors in (3.15) may be of theoretical interest, but for practical considerations, it is important that we can *represent* the entire matrix G by factorized matrices for admissible blocks and by small matrices for inadmissible leaves. To capture this concept, we introduce the *representation* of a hierarchical matrix explicitly, so that we can, e.g., investigate the storage requirements.

Definition 3.27 ($\mathcal{H}$-matrix representation) *Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a block tree, and let $G \in \mathcal{H}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, k)$ be a hierarchical matrix.*

Let $A = (A_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$, $B = (B_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$, and $N = (N_b)_{b\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}}$ be families of matrices. We call (A, B, N) an $\mathcal{H}$-matrix representation *of G if*

$$G|_{\hat{t}\times\hat{s}} = A_b B_b^*, \qquad A_b \in \mathbb{K}^{\hat{t}\times k},\ B_b \in \mathbb{K}^{\hat{s}\times k} \qquad \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},$$
$$G|_{\hat{t}\times\hat{s}} = N_b \qquad \text{for all } b = (t,s) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}.$$

In the case of the gravitational potential, (3.8) is of the form (3.15) if we let $k := (m+1)^d$. In the one-dimensional model problem, (2.8) and (2.16) are also of this form, but with $k = m$ and $k = m + 1$, respectively. This means that the three matrix approximations we have constructed so far are hierarchical matrices, and general statements about and algorithms for hierarchical matrices apply directly to them.

3.6 Complexity estimates

In our applications, $\mathcal{H}$-matrix approximations take the place of the original matrix G, therefore we have to store only these approximations and we have to perform arithmetic operations like the matrix-vector multiplication only with these approximations.

In order to obtain useful estimates for the storage requirements and the algorithmic complexity, we have to ensure

- that the inadmissible leaves of the block tree correspond to small matrices, and
- that the overall number of blocks is not too large.

For the first property, we can rely on the construction of the block tree that stops only once the row or column clusters are leaves. If the leaves are small, we immediately obtain a bound for the number of coefficients in the resulting $\mathcal{H}$-matrix representation.

Definition 3.28 (Admissible block tree) *Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a block tree, and let $\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ and $\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$ denote the admissible and inadmissible leaves of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$.*

The block tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is called admissible *if*

$$(t,s) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}} \Rightarrow (t \in \mathcal{L}_\mathcal{I} \vee s \in \mathcal{L}_\mathcal{J}) \qquad \text{holds for all } (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}},$$

and it is called strictly admissible *if*

$$(t,s) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}} \Rightarrow (t \in \mathcal{L}_\mathcal{I} \wedge s \in \mathcal{L}_\mathcal{J}) \qquad \text{holds for all } (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}.$$

Definition 3.29 (Cluster resolution) *Let $\mathcal{T}_\mathcal{I}$ be a cluster tree. We call*

$$r_\mathcal{I} := \max\{|\hat{t}| \ : \ t \in \mathcal{L}_\mathcal{I}\},$$

i.e., that maximal number of indices in leaf clusters, the resolution *of $\mathcal{T}_\mathcal{I}$.*

Lemma 3.30 (Storage requirements) *Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be an admissible block tree, let $r_\mathcal{I}$ and $r_\mathcal{J}$ denote the resolutions of $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$, and let $G \in \mathcal{H}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, k)$. An $\mathcal{H}$-matrix representation of G requires not more than*

$$\max\{k, r_\mathcal{I}, r_\mathcal{J}\} \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} (|\hat{t}| + |\hat{s}|)$$

units of storage.

Proof. Let $b = (t, s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$. If b is an admissible leaf, we have to store the matrices $A_b \in \mathbb{K}^{\hat{t}\times k}$ and $B_b \in \mathbb{K}^{\hat{s}\times k}$, and this takes

$$k(|\hat{t}| + |\hat{s}|)$$

units of storage.

If b is an inadmissible leaf, we have to store $N_b \in \mathbb{K}^{\hat{t}\times\hat{s}}$, and this takes $|\hat{t}|\,|\hat{s}|$ units of storage. Since $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is admissible, t or s has to be a leaf. In the first case we have $|\hat{t}| \le r_\mathcal{I}$ and can bound the storage requirements by

$$r_\mathcal{I}\,|\hat{s}| \le r_\mathcal{I}(|\hat{t}| + |\hat{s}|),$$

in the second case we have $|\hat{s}| \le r_\mathcal{J}$ and obtain the bound

$$r_\mathcal{J}\,|\hat{t}| \le r_\mathcal{J}(|\hat{t}| + |\hat{s}|).$$

Taking the maximum of the three estimates yields that each leaf $b = (t, s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$ requires not more than

$$\max\{k, r_\mathcal{I}, r_\mathcal{J}\}(|\hat{t}| + |\hat{s}|)$$

units of storage. Accumulating the requirements of all leaves leads to our estimate. ∎

In order to bound sums of the form

$$\sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} |\hat{t}|,$$

we employ the concept of sparse block trees [21]: Using admissibility conditions like (2.11) and (3.11), the row and column clusters of *inadmissible* blocks are geometrically close (cf. Figure 3.3). If we use a geometrically regular clustering strategy, the clusters have a certain size and do not (or hardly) overlap, and this means that only a bounded number of clusters can be close enough to make them inadmissible, the number of inadmissible blocks for a fixed row or column cluster is bounded.

Our strategy for constructing block trees ensures that only inadmissible blocks are split, therefore the parent of every block in $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is inadmissible, and we have just seen that the number of inadmissible blocks can be bounded.

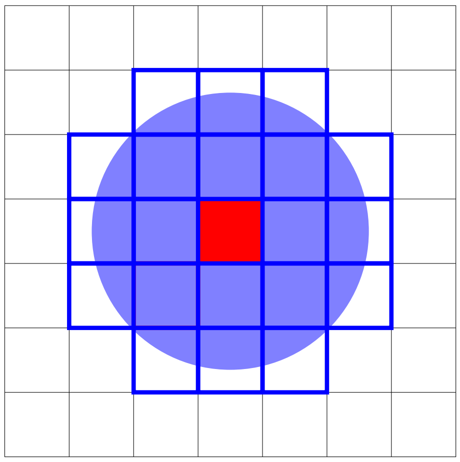

Figure 3.3: Given a cluster (red), only a bounded number of other clusters (blue) are close enough to lead to inadmissible blocks.

Definition 3.31 (Sparse block tree) *Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a block tree for the cluster trees $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$. We call*

$$\mathrm{row}(t) := \{s \in \mathcal{T}_{\mathcal{J}} \ :\ (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}\} \qquad\qquad \textit{for all } t \in \mathcal{T}_{\mathcal{I}}$$

the block rows *for the row clusters t and*

$$\mathrm{col}(s) := \{t \in \mathcal{T}_{\mathcal{I}} \ :\ (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}\} \qquad\qquad \textit{for all } s \in \mathcal{T}_{\mathcal{J}}$$

the block columns *for the column clusters s.*

Let $C_{\mathrm{sp}} \in \mathbb{N}$. We call $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ C_{sp}-sparse *if*

$$\begin{aligned} \#\,\mathrm{row}(t) &\leq C_{\mathrm{sp}} && \textit{for all } t \in \mathcal{T}_{\mathcal{I}}, \\ \#\,\mathrm{col}(s) &\leq C_{\mathrm{sp}} && \textit{for all } s \in \mathcal{T}_{\mathcal{J}}. \end{aligned}$$

If $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is sparse, we can use the sparsity constant C_{sp} to replace a sum of blocks by a sum of clusters, we have

$$\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} |\hat{t}| = \sum_{t\in\mathcal{T}_{\mathcal{I}}} \sum_{s\in\mathrm{row}(t)} |\hat{t}| \leq C_{\mathrm{sp}} \sum_{t\in\mathcal{T}_{\mathcal{I}}} |\hat{t}|.$$

In order to bound this sum, we can take advantage of the fact that Lemma 3.18 implies that an index $i \in \mathcal{I}$ appears at most once on each level of the cluster tree. As in Definition 3.15, we denote the *depth* of the cluster tree $\mathcal{T}_{\mathcal{I}}$ by

$$p_{\mathcal{I}} := \max\{\mathrm{level}(t) \ :\ t \in \mathcal{T}_{\mathcal{I}}\}.$$

Since the level numbers start at zero for the root, the total number of levels is $p_{\mathcal{I}} + 1$.

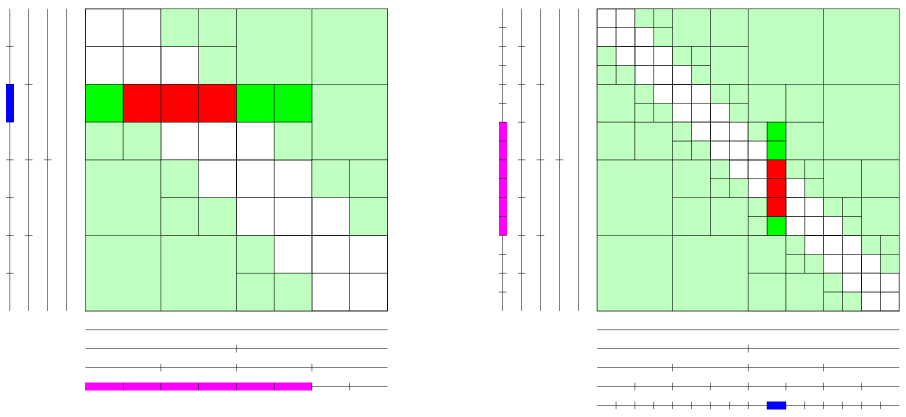

Figure 3.4: Block row and block column (magenta) for a given row or column cluster (blue)

Lemma 3.32 (Cluster sum) *Let $\mathcal{T}_\mathcal{I}$ be a cluster tree of depth $p_\mathcal{I}$. We have*

$$\sum_{t\in\mathcal{T}_\mathcal{I}} |\hat{t}| \leq (p_\mathcal{I}+1)|\mathcal{I}|, \qquad\qquad \sum_{t\in\mathcal{L}_\mathcal{I}} |\hat{t}| = |\mathcal{I}|.$$

Proof. Since Lemma 3.18 implies that the index sets for all clusters on the same level are disjoint, we have

$$\sum_{\substack{t\in\mathcal{T}_\mathcal{I}\\ \mathrm{level}(t)=\ell}} |\hat{t}| = \left| \bigcup_{\substack{t\in\mathcal{T}_\mathcal{I}\\ \mathrm{level}(t)=\ell}} \hat{t} \right| \leq |\mathcal{I}| \qquad\qquad \text{for all } \ell\in\mathbb{N}_0$$

and conclude

$$\sum_{t\in\mathcal{T}_\mathcal{I}} |\hat{t}| = \sum_{\ell=0}^{p_\mathcal{I}} \sum_{\substack{t\in\mathcal{T}_\mathcal{I}\\ \mathrm{level}(t)=\ell}} |\hat{t}| \leq \sum_{\ell=0}^{p_\mathcal{I}} |\mathcal{I}| = (p_\mathcal{I}+1)|\mathcal{I}|.$$

The second equation follows directly from Corollary 3.19. ∎

We could apply this result directly to find a bound for the storage requirements, but it is possible that the depth $p_{\mathcal{I}\times\mathcal{J}}$ of the block tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is smaller than the depths $p_\mathcal{I}$ and $p_\mathcal{J}$ of the corresponding cluster trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$, and in these situations a sharper estimate can be derived, since Definition 3.20 implies that the levels of the row and column clusters of a block cannot be larger than the level of the block.

Lemma 3.33 (Block level) *Let $b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$. We have*

$$\mathrm{level}(b) = \max\{\mathrm{level}(t), \mathrm{level}(s)\}.$$

If $\mathrm{level}(t) < \mathrm{level}(b)$*, the row cluster t is a leaf. If* $\mathrm{level}(s) < \mathrm{level}(b)$*, the column cluster s is a leaf.*

Proof. By induction on $\operatorname{level}(b)$.

Let $b \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with $\operatorname{level}(b) = 0$. Then b is the root of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, and Definition 3.20 yields that t and s are the roots of $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$, respectively, so we have $\operatorname{level}(t) = \operatorname{level}(s) = 0$.

Let now $n \in \mathbb{N}_0$ be given such that $\operatorname{level}(b) = \max\{\operatorname{level}(t), \operatorname{level}(s)\}$ holds for all blocks $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with $\operatorname{level}(b) = n$, that $\operatorname{level}(t) < \operatorname{level}(b)$ implies $t \in \mathcal{L}_{\mathcal{I}}$, and that $\operatorname{level}(s) < \operatorname{level}(b)$ implies $s \in \mathcal{L}_{\mathcal{J}}$.

Let $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with $\operatorname{level}(b) = n+1$. Since $\operatorname{level}(b) > 0$, we can find a parent block $b_+ = (t_+, s_+) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with $b \in \operatorname{chil}(b_+)$, and Definition 3.15 yields $\operatorname{level}(b_+) = n$.

If $\operatorname{level}(t_+) < \operatorname{level}(b_+)$, we have $t_+ \in \mathcal{L}_{\mathcal{I}}$ by the induction assumption. Definition 3.20 implies $t = t_+$ and $s \in \operatorname{chil}(s_+)$, and we can use the induction assumption to obtain

$$\operatorname{level}(b) = \operatorname{level}(b_+) + 1 = \operatorname{level}(s_+) + 1 = \operatorname{level}(s).$$

Together with $\operatorname{level}(t) = \operatorname{level}(t_+) < \operatorname{level}(b_+)$, this proves our claim.

If $\operatorname{level}(s_+) < \operatorname{level}(b_+)$, we can use the same arguments.

Otherwise we have $\operatorname{level}(t_+) = \operatorname{level}(s_+) = \operatorname{level}(b_+)$. Definition 3.20 guarantees $\operatorname{chil}(t_+) \neq \emptyset$ or $\operatorname{chil}(s_+) \neq \emptyset$. We have

$$\operatorname{level}(t) = \begin{cases} \operatorname{level}(t_+) + 1 = \operatorname{level}(b_+) + 1 = \operatorname{level}(b) & \text{if } \operatorname{chil}(t_+) \neq \emptyset, \\ \operatorname{level}(t_+) \qquad < \operatorname{level}(b_+) + 1 = \operatorname{level}(b) & \text{otherwise,} \end{cases}$$

$$\operatorname{level}(s) = \begin{cases} \operatorname{level}(s_+) + 1 = \operatorname{level}(b_+) + 1 = \operatorname{level}(b) & \text{if } \operatorname{chil}(s_+) \neq \emptyset, \\ \operatorname{level}(s_+) \qquad < \operatorname{level}(b_+) + 1 = \operatorname{level}(b) & \text{otherwise,} \end{cases}$$

and this completes the proof. ∎

Lemma 3.34 (Block sum) *Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be an admissible C_{sp}-sparse block tree of depth $p_{\mathcal{I}\times\mathcal{J}}$. We have*

$$\sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} |\hat{t}| \le C_{\mathrm{sp}}(p_{\mathcal{I}\times\mathcal{J}} + 1)|\mathcal{I}|, \qquad \sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} |\hat{s}| \le C_{\mathrm{sp}}(p_{\mathcal{I}\times\mathcal{J}} + 1)|\mathcal{J}|.$$

Proof. Lemma 3.33 yields

$$\operatorname{level}(t) \le \operatorname{level}(b), \qquad \operatorname{level}(s) \le \operatorname{level}(b) \qquad \text{for all } b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}},$$

and we find

$$\sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} |\hat{t}| = \sum_{\substack{t\in\mathcal{T}_{\mathcal{I}} \\ \operatorname{level}(t)\le p_{\mathcal{I}\times\mathcal{J}}}} \sum_{s\in\operatorname{row}(t)} |\hat{t}| \le C_{\mathrm{sp}} \sum_{\substack{t\in\mathcal{T}_{\mathcal{I}} \\ \operatorname{level}(t)\le p_{\mathcal{I}\times\mathcal{J}}}} |\hat{t}| = C_{\mathrm{sp}} \sum_{\ell=0}^{p_{\mathcal{I}\times\mathcal{J}}} \sum_{\substack{t\in\mathcal{T}_{\mathcal{I}} \\ \operatorname{level}(t)=\ell}} |\hat{t}|.$$

We can again use Lemma 3.18 to obtain

$$\sum_{\substack{t\in\mathcal{T}_{\mathcal{I}} \\ \operatorname{level}(t)=\ell}} |\hat{t}| = \Bigg| \bigcup_{\substack{t\in\mathcal{T}_{\mathcal{I}} \\ \operatorname{level}(t)=\ell}} \hat{t} \Bigg| \le |\mathcal{I}| \qquad \text{for all } \ell \in \mathbb{N}_0,$$

and we conclude

$$\sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} |\hat{t}| \le C_{\mathrm{sp}}(p_{\mathcal{I}\times\mathcal{J}}+1)|\mathcal{I}|.$$

Similar arguments can be applied for the second estimate. ∎

Theorem 3.35 (Storage requirements) *Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be an admissible C_{sp}-sparse block tree of depth $p_{\mathcal{I}\times\mathcal{J}}$. Let $r_{\mathcal{I}}$ and $r_{\mathcal{J}}$ be the resolutions of the row and column cluster trees $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$. A corresponding $\mathcal{H}$-matrix representation of a matrix G with local rank k requires not more than*

$$C_{\mathrm{sp}} \max\{k, r_{\mathcal{I}}, r_{\mathcal{J}}\}(p_{\mathcal{I}\times\mathcal{J}}+1)(|\mathcal{I}|+|\mathcal{J}|)$$

units of storage.

Proof. We combine Lemma 3.30 and Lemma 3.34 to find that not more than

$$\begin{aligned}
\max\{k, r_{\mathcal{I}}, r_{\mathcal{J}}\} & \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} (|\hat{t}|+|\hat{s}|) \\
&= \max\{k, r_{\mathcal{I}}, r_{\mathcal{J}}\} \left(\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} |\hat{t}| + \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} |\hat{s}| \right) \\
&\le \max\{k, r_{\mathcal{I}}, r_{\mathcal{J}}\} \left(\sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} |\hat{t}| + \sum_{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}} |\hat{s}| \right) \\
&\le \max\{k, r_{\mathcal{I}}, r_{\mathcal{J}}\} \left(C_{\mathrm{sp}}(p_{\mathcal{I}\times\mathcal{J}}+1)|\mathcal{I}| + C_{\mathrm{sp}}(p_{\mathcal{I}\times\mathcal{J}}+1)|\mathcal{J}| \right) \\
&= C_{\mathrm{sp}} \max\{k, r_{\mathcal{I}}, r_{\mathcal{J}}\}(p_{\mathcal{I}\times\mathcal{J}}+1)(|\mathcal{I}|+|\mathcal{J}|)
\end{aligned}$$

units of storage are required. ∎

Remark 3.36 (Matrix-vector multiplication) *Given an $\mathcal{H}$-matrix G, the matrix-vector multiplication $y \leftarrow y + Gx$ can be performed blockwise, i.e., by taking the steps*

$$y|_{\hat{t}} \leftarrow y|_{\hat{t}} + G|_{\hat{t}\times\hat{s}} x|_{\hat{s}} \qquad \text{for all } b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}$$

in any order. For admissible leaves, computing $z \leftarrow B_b^* x|_{\hat{s}}$ takes $(2k-1)|\hat{s}| \le 2k|\hat{s}|$ operations, and the update $y|_{\hat{t}} \leftarrow y|_{\hat{t}} + A_b z$ takes $2k|\hat{t}|$ operations.

For inadmissible leaves, we have $2|\hat{t}|\,|\hat{s}|$ operations. Assuming that $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is admissible, we can bound this by $2\max\{r_{\mathcal{I}}, r_{\mathcal{J}}\}(|\hat{t}|+|\hat{s}|)$ and obtain the upper bound

$$2\max\{k, r_{\mathcal{I}}, r_{\mathcal{J}}\} \sum_{\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} |\hat{t}|+|\hat{s}|$$

for the number of arithmetic operations. This is the same sum as in Lemma 3.30, and we can proceed as before to obtain the bound

$$2C_{\mathrm{sp}} \max\{k, r_{\mathcal{I}}, r_{\mathcal{J}}\}(p_{\mathcal{I}\times\mathcal{J}}+1)(|\mathcal{I}|+|\mathcal{J}|)$$

for the number of operations.

3.7 Estimates for cluster trees and block trees

Our complexity estimates rely on a number of assumptions:

1. The block tree is admissible.

   This can be guaranteed by continuing to subdivide inadmissible blocks as long as the row or the column cluster has children.

2. The resolution of the cluster trees is bounded.

   This can be guaranteed by continuing to subdivide clusters as long as they contain too many indices.

3. The block tree is sparse.

   Proving this assumption can be quite challenging. Here, we focus on very regular cluster strategies and a simple admissibility condition.

4. The depth of the block tree is bounded.

   We can handle this assumption by proving that the depths of the cluster trees are bounded and applying Lemma 3.33.

We briefly recall three popular cluster strategies. All are based on assigning points $x_i \in \mathbb{R}^d$ to all indices $i \in \mathcal{I}$, e.g., the positions of planets or the nodal points of finite element basis functions, and splitting the "point clouds" corresponding to clusters to construct children.

We prescribe a resolution $r_\mathcal{I} \in \mathbb{N}$ and we keep splitting clusters recursively until the condition $|\hat{t}| \leq r_\mathcal{I}$ is met.

Adaptive bisection Clusters correspond to axis-parallel boxes

$$t = [a_1, b_1] \times \cdots \times [a_d, b_d].$$

We choose the direction of maximal extent $\iota \in [1 : d]$, i.e., we have

$$b_\iota - a_\iota \geq b_\kappa - a_\kappa \qquad \text{for all } \kappa \in [1 : d].$$

If this condition does not lead to a unique choice, we pick the smallest ι.

We denote the midpoint of the ι-th interval by

$$c_\iota := \frac{b_\iota + a_\iota}{2}$$

and construct two children

$$t_1 := \{x \in t \ : \ x_\iota \leq c_\iota\}, \qquad t_2 := \{x \in t \ : \ x_\iota \geq c_\iota\}.$$

The corresponding index sets are

$$\hat{t}_1 := \{i \in \hat{t} \ : \ x_{i,\iota} \leq c_\iota\}, \qquad \hat{t}_2 := \{i \in \hat{t} \ : \ x_{i,\iota} > c_\iota\},$$

where we arbitrarily assign indices on the boundary between the two children to t_1.

Regular bisection We proceed as in the adaptive bisection approach, only instead of choosing the direction of maximal extent, we cycle through all directions $\iota \in [1 : d]$.

Tensor bisection Clusters correspond to axis-parallel boxes

$$t = [a_1, b_1] \times \cdots \times [a_d, b_d]$$

and are split along *all* coordinate axes simultaneously.

We denote the midpoints by

$$c_\iota := \frac{b_\iota + a_\iota}{2} \qquad \text{for all } \iota \in [1 : d]$$

and define

$$a_\iota^{(0)} := a_\iota, \qquad b_\iota^{(0)} := c_\iota, \qquad a_\iota^{(1)} := c_\iota, \qquad b_\iota^{(1)} := b_\iota \qquad \text{for all } \iota \in [1 : d].$$

This implies

$$\begin{aligned} &[a_\iota^{(0)}, b_\iota^{(0)}] \cup [a_\iota^{(1)}, b_\iota^{(1)}] = [a_\iota, b_\iota], \\ &[a_\iota^{(0)}, b_\iota^{(0)}] \cap [a_\iota^{(1)}, b_\iota^{(1)}] = \{c_\iota\} \qquad \text{for all } \iota \in [1 : d], \end{aligned}$$

i.e., we split all intervals $[a_\iota, b_\iota]$ along the middle into $[a_\iota^{(0)}, b_\iota^{(0)}]$ and $[a_\iota^{(1)}, b_\iota^{(1)}]$.

Given the multiindex set $S := \{0, 1\}^d$, we construct 2^d children

$$t_\nu := [a_1^{(\nu_1)}, b_1^{(\nu_1)}] \times \cdots \times [a_d^{(\nu_d)}, b_d^{(\nu_d)}] \qquad \text{for all } \nu \in S.$$

The corresponding index sets are given by

$$\begin{aligned} \hat{t}_\nu := \{i \in \hat{t} \ : \ \forall \iota \in [1 : d] \ : \ &(\nu_\iota = 0 \Rightarrow x_{i,\iota} \le c_\iota) \wedge \\ &(\nu_\iota = 1 \Rightarrow x_{i,\iota} > c_\iota)\} \qquad \text{for all } \nu \in S. \end{aligned}$$

Remark 3.37 (Diameters) *For the tensor bisection strategy, we immediately obtain*

$$\operatorname{diam}(t') \le \frac{\operatorname{diam}(t)}{2} \qquad \textit{for all } t' \in \operatorname{chil}(t),\ t \in \mathcal{T}_{\mathcal{I}}.$$

For the regular bisection strategy, we have to take d splitting steps before the diameter is halved, i.e.,

$$\operatorname{diam}(t^*) \le \frac{\operatorname{diam}(t)}{2} \qquad \textit{for all } t^* \in \operatorname{desc}(t),\ \operatorname{level}(t^*) = \operatorname{level}(t) + d,\ t \in \mathcal{T}_{\mathcal{I}}.$$

Adaptive bisection is a little more challenging. We again consider a cluster $t \in \mathcal{T}_{\mathcal{I}}$ and perform d splitting steps to obtain a descendant $t^* \in \operatorname{desc}(t)$ with $\operatorname{level}(t^*) = \operatorname{level}(t) + d$. We let

$$t^* = [a_1^*, b_1^*] \times \cdots \times [a_d^*, b_d^*].$$

For every $\iota \in [1:d]$, let $\nu_\iota \in [0:d]$ be the number of times we have split clusters in the ι-th direction. Since we have performed d steps, we have

$$|\nu| = \nu_1 + \ldots + \nu_d = d.$$

We choose a direction $\iota \in [1:d]$ with $\nu_\iota > 0$. Our strategy splits only along the direction of maximal extent. The ι-th direction can only have been chosen ν_ι times if it was still the direction of maximal extent after $\nu_\iota - 1$ splits, i.e.,

$$2^{\nu_\iota - 1}(b^*_\iota - a^*_\iota) \geq b^*_\kappa - a^*_\kappa \qquad \text{for all } \kappa \in [1:d].$$

We conclude

$$\begin{aligned}\operatorname{diam}(t^*) &= \max\{b^*_\kappa - a^*_\kappa \ :\ \kappa \in [1:d]\} \leq 2^{\nu_\iota - 1}(b^*_\iota - a^*_\iota) \\ &= \frac{2^{\nu_\iota}(b^*_\iota - a^*_\iota)}{2} = \frac{b_\iota - a_\iota}{2} \leq \frac{\operatorname{diam}(t)}{2},\end{aligned}$$

i.e., adaptive bisection reduces diameters at least as quickly as regular bisection.

Now that we know how many levels are required to reduce the diameter of a cluster, we can easily calculate how many levels are required until clusters contain only one point, i.e., when the splitting procedure stops.

Lemma 3.38 (Depth) *Let $H \in \mathbb{R}_{>0}$ be the diameter of the root cluster. Let*

$$h := \min\{\|x_i - x_j\|_\infty \ :\ i,j \in \mathcal{I},\ i \neq j\}$$

be the minimal distance between two points. The depth of the cluster tree is bounded by

$$p_\mathcal{I} \leq \begin{cases} \lfloor \log_2(H/h) \rfloor + 1 & \text{for tensor bisection,} \\ d(\lfloor \log_2(H/h) \rfloor + 1) & \text{for regular and adaptive bisection.} \end{cases}$$

Proof. We first consider the tensor bisection strategy. Using

$$\operatorname{diam}(t') \leq \frac{\operatorname{diam}(t)}{2} \qquad \text{for all } t \in \mathcal{T}_\mathcal{I},\ t' \in \operatorname{chil}(t),$$

a simple induction yields

$$\operatorname{diam}(t) \leq 2^{-\ell} H \qquad \text{for all } t \in \mathcal{T}_\mathcal{I},\ \operatorname{level}(t) = \ell.$$

If we use $\ell := \lfloor \log_2(H/h) \rfloor + 1$, we have

$$2^{-\ell} < 2^{-\log_2(H/h)} = h/H$$

and therefore $\operatorname{diam}(t) < h$ for $t \in \mathcal{T}_\mathcal{I}$ with $\operatorname{level}(t) = \ell$. Due to our choice of h, this means that t can contain at most one point and therefore has to be a leaf.

For regular and adaptive bisection, we can use

$$\operatorname{diam}(t^*) \leq \frac{\operatorname{diam}(t)}{2} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}},\ t^* \in \operatorname{desc}(t),\ \operatorname{level}(t^*) - \operatorname{level}(t) = d,$$

to obtain

$$\operatorname{diam}(t) \leq 2^{-\ell} H \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}},\ \operatorname{level}(t) = \ell d.$$

Now we can proceed as before. ■

Once we have constructed cluster trees $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$, we can consider the block tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$. Definitions 3.20 and 3.28 already prescribe the essential steps by which a minimal admissible block tree can be constructed: we start with the root block $r = (\operatorname{root}(\mathcal{T}_{\mathcal{I}}), \operatorname{root}(\mathcal{T}_{\mathcal{J}}))$. If a block $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is admissible, we stop and make it a leaf. If t and s are leaves, we also stop and make b an inadmissible leaf. Otherwise we subdivide b into its sons

$$\operatorname{sons}(b) = \begin{cases} \operatorname{sons}(t) \times \operatorname{sons}(s) & \text{if } \operatorname{sons}(t) \neq \emptyset,\ \operatorname{sons}(s) \neq \emptyset, \\ \{t\} \times \operatorname{sons}(s) & \text{if } \operatorname{sons}(t) = \emptyset,\ \operatorname{sons}(s) \neq \emptyset, \\ \operatorname{sons}(t) \times \{s\} & \text{if } \operatorname{sons}(t) \neq \emptyset,\ \operatorname{sons}(s) = \emptyset. \end{cases}$$

and treat each of these sons $b' \in \operatorname{sons}(b)$ recursively.

For our complexity analysis, we not only need $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ to be admissible, it also has to be sparse. In order to investigate this property, we follow the approach outlined in [21]: a block $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is only constructed if its father $b^+ = (t^+, s^+) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is not admissible. Since our cluster construction ensures that no cluster has more than 2^d sons for tensor bisection and two sons for adaptive or regular bisection, bounding the number of inadmissible blocks yields an estimate for the number of blocks.

A bound for the number of inadmissible blocks can be obtained by counting how many clusters "fit into a ball" surrounding a given cluster. If the ball centered at the cluster t is sufficiently large, it contains all clusters s such that (t, s) is inadmissible, since any cluster intersecting the outside of the sphere has to be admissible.

In order to turn this into a precise mathematical proof, we require that the clusters' aspect ratios do not degenerate.

Lemma 3.39 (Inadmissible blocks) *Let $C_{\mathrm{ar}} \in \mathbb{R}_{>0}$ be given with*

$$\operatorname{diam}(t)^d \leq C_{\mathrm{ar}} \lambda_d(t) \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}},$$

where λ_d denotes the d-dimensional Lebesgue measure. Let

$$\omega_d := \lambda_d(\{x \in \mathbb{R}^d \ :\ \|x\|_2 \leq 1\})$$

denote the measure of the d-dimensional unit ball. Let $t \in \mathcal{T}_{\mathcal{I}}$. The set of inadmissible clusters

$$\mathcal{I}_t := \{s \in \mathcal{T}_{\mathcal{I}} \ :\ \operatorname{level}(s) = \operatorname{level}(t),\ 2\eta \operatorname{dist}(t, s) < \operatorname{diam}(t)\}$$

is bounded by

$$|\mathcal{I}_t| \leq C_{\mathrm{ar}} \omega_d \frac{(3\sqrt{d} + 1/\eta)^d}{2^d}.$$

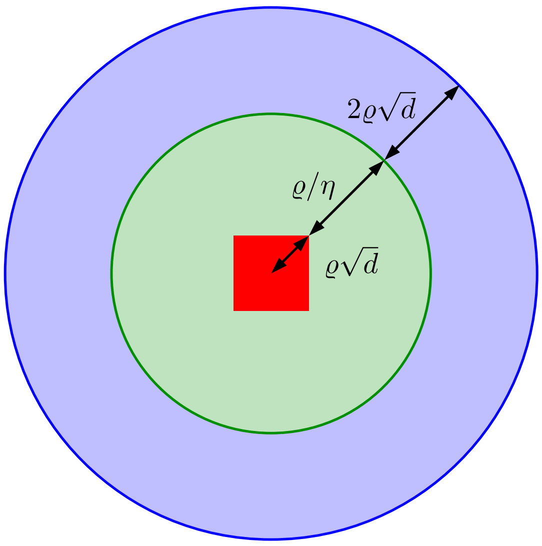

Figure 3.5: Inadmissible clusters intersect the green ball and are in the blue one.

Proof. Let $\ell := \operatorname{level}(t)$, and let $m_t \in t$ denote the center of the axis-parallel box t. We let $\varrho := \operatorname{diam}(t)/2$ and observe

$$\|x - m\|_2^2 = \sum_{\iota=1}^{d} (x_\iota - m_\iota)^2 \leq \sum_{\iota=1}^{d} \frac{\operatorname{diam}(t)^2}{4} = d\frac{\operatorname{diam}(t)^2}{4} = d\varrho^2 \qquad \text{for all } x \in t.$$

We prove that all inadmissible clusters $s \in \mathcal{I}_t$ are contained in the ball

$$\mathcal{B} := \{y \in \mathbb{R}^d \ : \ \|y - m\|_2 \leq (3\sqrt{d} + 1/\eta)\varrho\},$$

cf. Figure 3.5. Let $s \in \mathcal{I}_t$, i.e., we have

$$\operatorname{dist}(t, s) < \frac{\operatorname{diam}(t)}{2\eta} = \frac{\varrho}{\eta}.$$

In order to prove $s \subseteq \mathcal{B}$, we let $z \in s$ and have to show $z \in \mathcal{B}$. Let $x \in t$ and $y \in s$ with $\|x - y\|_2 = \operatorname{dist}(t, s)$. We find

$$\|z - y\|_2^2 = \sum_{\iota=1}^{d} (z_\iota - y_\iota)^2 \leq \sum_{\iota=1}^{d} \operatorname{diam}(s)^2 = d \operatorname{diam}(s)^2 = 4d\varrho^2$$

due to $\operatorname{diam}(s) = \operatorname{diam}(t) = 2\varrho$ and find

$$\begin{aligned} \|z - m\|_2 &= \|z - y + y - x + x - m\|_2 \leq \|z - y\|_2 + \|y - x\|_2 + \|x - m\|_2 \\ &\leq 2\sqrt{d}\varrho + \operatorname{dist}(t, s) + \sqrt{d}\varrho < 3\sqrt{d}\varrho + \frac{\varrho}{\eta} = (3\sqrt{d} + 1/\eta)\varrho, \end{aligned}$$

so we have indeed proven $z \in \mathcal{B}$.

By our construction, the clusters on the same level as t are disjoint up to null sets and have the same measure as t, so we have

$$\begin{aligned}|\mathcal{I}_t| &= \frac{1}{\lambda_d(t)} \sum_{s \in \mathcal{I}_t} \lambda_d(s) = \frac{\lambda_d\left(\bigcup_{s \in \mathcal{I}_t} s\right)}{\lambda_d(t)} \leq \frac{\lambda_d(\mathcal{B})}{\lambda_d(t)} = \frac{\omega_d((3\sqrt{d}+1/\eta)\varrho)^d}{\lambda_d(t)} \\ &\leq C_{\mathrm{ar}}\omega_d \frac{(3\sqrt{d}+1/\eta)^d \varrho^d}{\operatorname{diam}(t)^d} = C_{\mathrm{ar}}\omega_d \frac{(3\sqrt{d}+1/\eta)^d \varrho^d}{2^d \varrho^d} = C_{\mathrm{ar}}\omega_d \frac{(3\sqrt{d}+1/\eta)^d}{2^d}.\end{aligned}$$

■

Lemma 3.40 (Sparsity) *Let $\mathcal{T}_{\mathcal{I}}$ be a cluster tree constructed by our algorithm. $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ be a minimal admissible block tree for the admissibility condition (3.11) with $\eta \in \mathbb{R}_{>0}$. The block tree $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ is C_{sp}-sparse with*

$$C_{\mathrm{sp}} := \begin{cases} C_{\mathrm{ar}}\omega_d(3\sqrt{d}+1/\eta)^d & \text{for tensor bisection,} \\ C_{\mathrm{ar}}\omega_d(3\sqrt{d}+1/\eta)^d 2^{1-d} & \text{for regular and adaptive bisection,} \end{cases}$$

where C_{ar} and ω_d are defined as in Lemma 3.39.

Proof. Since $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ is the minimal *admissible* block tree, blocks are not subdivided as soon as the row *or* the column cluster is a leaf. Together with Definition 3.20, this implies $\operatorname{level}(t) = \operatorname{level}(s)$ for all $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{I}}$.

Let $t \in \mathcal{T}_{\mathcal{I}}$. If $t = \operatorname{root}(\mathcal{T}_{\mathcal{I}})$, $(t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ already implies $t = s$.

Otherwise let $t^+ \in \mathcal{T}_{\mathcal{I}}$ denote the parent of t.

Let $s \in \operatorname{row}(t)$, and let $s^+ \in \mathcal{T}_{\mathcal{I}}$ denote the parent of s. Due to the minimality of the block tree, (t^+, s^+) cannot be admissible, since otherwise the algorithm would have stopped before creating (t,s).

Due to Lemma 3.39, we have

$$|\mathcal{I}_{t^+}| \leq C_{\mathrm{ar}}\omega_d \frac{(3\sqrt{d}+1/\eta)^d}{2^d}.$$

For the tensor bisection strategy, s^+ can have no more than 2^d children, and we conclude

$$\begin{aligned}|\operatorname{row}(t)| &= |\{s \in \mathcal{T}_{\mathcal{I}} \ : \ (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{I}}\}| \leq \Big| \bigcup_{s^+ \in \mathcal{I}_{t^+}} \operatorname{chil}(s^+) \Big| \\ &\leq 2^d |\mathcal{I}_{t^+}| \leq 2^d C_{\mathrm{ar}}\omega_d \frac{(3\sqrt{d}+1/\eta)^d}{2^d} = C_{\mathrm{ar}}\omega_d(3\sqrt{d}+1/\eta)^d.\end{aligned}$$

For the regular or adaptive bisection strategy, s^+ can have no more than two children, and we can modify the argument to obtain

$$\begin{aligned}|\operatorname{row}(t)| &= |\{s \in \mathcal{T}_{\mathcal{I}} \ : \ (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{I}}\}| \leq \Big| \bigcup_{s^+ \in \mathcal{I}_{t^+}} \operatorname{chil}(s^+) \Big| \\ &\leq 2|\mathcal{I}_{t^+}| \leq 2C_{\mathrm{ar}}\omega_d \frac{(3\sqrt{d}+1/\eta)^d}{2^d} = C_{\mathrm{ar}}\omega_d(3\sqrt{d}+1/\eta)^d 2^{1-d}.\end{aligned}$$

■

3.8 Improved interpolation error estimates*

The estimate provided by Corollary 3.10 for the interpolation error is not optimal: it requires the relative distance between the clusters t and s to be "large enough", and we have already seen in Section 2.8 that one-dimensional interpolation converges as long as the distance is non-zero. Since tensor interpolation is closely related to one-dimensional interpolation, we can expect this property to carry over to the multi-dimensional case.

To investigate the interpolation error a little closer, we fix a cluster

$$t = [a_1, b_1] \times \cdots \times [a_d, b_d]$$

and a closed set $s \subseteq \mathbb{R}^d$ with $\operatorname{dist}(t, s) > 0$.

We have already seen that it is sufficient to take a look at the error $f - \mathfrak{I}_{t,\iota}[f]$ of the partial interpolation operators for all direction indices $\iota \in [1 : d]$.

Lemma 3.41 (Best approximation) *Let $y \in s$, $\iota \in [1 : d]$, and*

$$f \colon t \to \mathbb{R}, \qquad x \mapsto g(x, y).$$

Let $x \in t$ and

$$\hat{f}_\iota \colon [-1, 1] \to \mathbb{R}, \qquad \tau \mapsto f(x_1, \ldots, x_{\iota-1}, \Phi_{[a_\iota, b_\iota]}(\tau), x_{\iota+1}, \ldots, x_d).$$

Using the Lebesgue stability constant Λ_m introduced in (2.24), we have

$$|f(x) - \mathfrak{I}_{t,\iota}[f](x)| \leq (1 + \Lambda_m) \|\hat{f}_\iota - p\|_{\infty,[-1,1]} \qquad \text{for all } p \in \Pi_m.$$

Proof. As in Lemma 3.5, we introduce the auxiliary function

$$f_\iota \colon [a_\iota, b_\iota] \to \mathbb{R}, \qquad z \mapsto f(x_1, \ldots, x_{\iota-1}, z, x_{\iota+1}, \ldots, x_d),$$

and recall

$$\mathfrak{I}_{t,\iota}[f](x) = \mathfrak{I}_{[a_\iota, b_\iota]}[f_\iota](x_\iota).$$

Due to the definition of the transformed interpolation operator $\mathfrak{I}_{[a_\iota, b_\iota]}$, we have

$$\begin{aligned} |f(x) - \mathfrak{I}_{t,\iota}[f](x)| &= |f_\iota(x_\iota) - \mathfrak{I}_{[a_\iota, b_\iota]}[f_\iota](x_\iota)| \\ &= |f_\iota \circ \Phi_{[a_\iota, b_\iota]}(\tau) - \mathfrak{I}[f_\iota \circ \Phi_{[a_\iota, b_\iota]}](\tau)| = |\hat{f}_\iota(\tau) - \mathfrak{I}[\hat{f}_\iota](\tau)| \end{aligned} \tag{3.16}$$

with $\tau := \Phi^{-1}_{[a_\iota, b_\iota]}(x_\iota) \in [-1, 1]$.

Let $p \in \Pi_m$. Due to $\mathfrak{I}[p] = p$, we have

$$\begin{aligned} \|\hat{f}_\iota - \mathfrak{I}[\hat{f}_\iota]\|_{\infty,[-1,1]} &= \|\hat{f}_\iota - p + \mathfrak{I}[p - \hat{f}_\iota]\|_{\infty,[-1,1]} \\ &\leq \|\hat{f}_\iota - p\|_{\infty,[-1,1]} + \|\mathfrak{I}[\hat{f}_\iota - p]\|_{\infty[-1,1]} \\ &\leq \|\hat{f}_\iota - p\|_{\infty,[-1,1]} + \Lambda_m \|\hat{f}_\iota - p\|_{\infty,[-1,1]} \\ &= (1 + \Lambda_m) \|\hat{f}_\iota - p\|_{\infty,[-1,1]}. \end{aligned}$$

Combining this estimate with (3.16) completes the proof. ■

Using this result, we "only" have to prove that $\hat{f}_\iota$ can be approximated by polynomials. We approach this task by rewriting the function.

Lemma 3.42 (Interpolant) *Let $\iota \in [1:d]$, let $x \in t$ and $y \in s$. We define vectors $m, p \in \mathbb{R}^d$ with*

$$m_\kappa := \begin{cases} \frac{b_\iota + a_\iota}{2} - y_\iota & \text{if } \kappa = \iota, \\ x_\kappa - y_\kappa & \text{otherwise}, \end{cases}$$

$$p_\kappa := \begin{cases} \frac{b_\iota - a_\iota}{2} & \text{if } \kappa = \iota, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } \kappa \in [1:d].$$

We have $\|p\|_2 \leq \operatorname{diam}(t)/2$, and

$$\|m + \tau p\|_2 \geq \operatorname{dist}(t,s), \qquad \hat{f}_\iota(\tau) = \frac{\gamma}{\|m + \tau p\|_2} \qquad \text{for all } \tau \in [-1,1].$$

Proof. We introduce the extension operator

$$E\colon [-1,1] \to t, \qquad \tau \mapsto (x_1, \ldots, x_{\iota-1}, \Phi_{[a_\iota, b_\iota]}(\tau), x_{\iota+1}, \ldots, x_d),$$

and write $\hat{f}_\iota$ in the form

$$\hat{f}_\iota(\tau) = g(E(\tau), y) = \frac{\gamma}{\|E(\tau) - y\|_2} \qquad \text{for all } \tau \in [-1,1].$$

Let $\tau \in [-1,1]$ Due to

$$\Phi_{[a_\iota, b_\iota]}(\tau) = \frac{b_\iota + a_\iota}{2} + \tau \frac{b_\iota - a_\iota}{2},$$

we find

$$E(\tau) = E(0) + \tau p, \qquad E(\tau) - y = E(0) - y + \tau p = m + \tau p.$$

Due to $E(\tau) \in t$, we have

$$\begin{aligned} \|m + \tau p\|_2 &= \|E(\tau) - y\|_2 \geq \operatorname{dist}(t,s), \\ \hat{f}_\iota(\tau) &= g(E(\tau), y) = \frac{\gamma}{\|E(\tau) - y\|_2} = \frac{\gamma}{\|m + \tau p\|_2}. \end{aligned}$$

∎

In the following, we fix $\iota \in [1:d]$ and write $\hat{f}$ instead of $\hat{f}_\iota$. We have already seen in Lemma 3.8 that we can find a complex number $w \in \mathbb{C}$ with $|w| = \|m\|_2$ such that

$$z \mapsto \frac{\gamma}{\sqrt{(w + zr)(\bar{w} + zr)}} \qquad \text{for all } z \in \mathbb{C},\ |z| < \zeta,$$

is a holomorphic extension of $\hat{f}$, where $r := \|p\|_2$ and $\zeta := \|m\|_2/r$.

In order to apply Theorem 2.29, we require a holomorphic extension of $\hat{f}$ not just to a disc around zero, but to a Bernstein disc that is as large as possible.

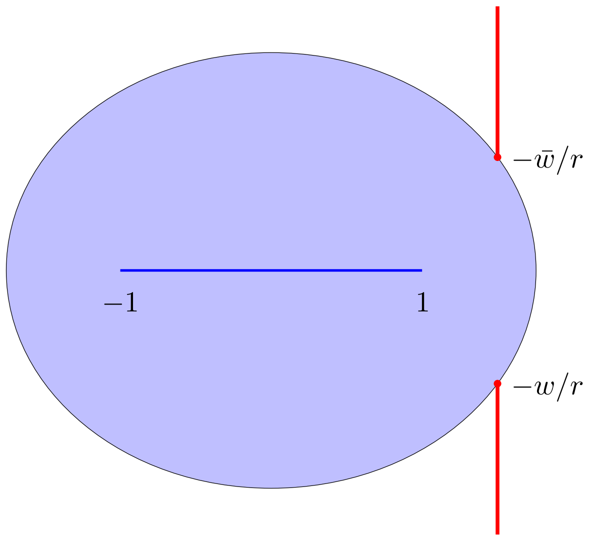

Figure 3.6: Maximal Bernstein disc $\mathcal{D}_\varrho$ for $w = -\frac{3}{2} + \iota\frac{3}{4}$ and $r = 1$

Lemma 3.43 (Maximal holomorphic extension) *Let $w = w_r + \iota w_i \in \mathbb{C} \setminus [-1, 1]$ with $w_r, w_i \in \mathbb{R}$ and*

$$\mathcal{S}_w := \mathbb{C} \setminus \{a + \iota b \ : \ a, b \in \mathbb{R} \text{ with } ar + w_r = 0 \text{ and } |b|\, r \geq |w_i|\}.$$

The function

$$\hat{f} \colon \mathcal{S}_w \to \mathbb{C}, \qquad\qquad z \mapsto \frac{\gamma}{\sqrt{(w + zr)(\bar{w} + zr)}},$$

is holomorphic.

Proof. We have to prove that $z \in \mathcal{S}_w$ implies that $(w + zr)(\bar{w} + zr)$ is contained in the domain $\mathbb{C} \setminus \mathbb{R}_{\leq 0}$ of the principal branch of the holomorphic square root function.

We accomplish this by contraposition: let $z \in \mathbb{C}$ be given with

$$(w + zr)(\bar{w} + zr) \in \mathbb{R}_{\leq 0}.$$

We will prove $z \notin \mathcal{S}_w$. Let $a, b \in \mathbb{R}$ with $z = a + \iota b$. We have

$$\begin{aligned}(w + zr)(\bar{w} + zr) &= (ar + w_r + \iota(br + w_i))(ar + w_r + \iota(br - w_i)) \\ &= (ar + w_r)^2 - (br + w_i)(br - w_i) + \iota(ar + w_r)(br + w_i + br - w_i) \\ &= (ar + w_r)^2 + w_i^2 - b^2r^2 + 2\iota(ar + w_r)br.\end{aligned}$$

Due to our assumption $(w + zr)(\bar{w} + zr) \in \mathbb{R}_{\leq 0}$, we have

$$(ar + w_r)br = 0, \qquad\qquad (ar + w_r)^2 + w_i^2 - b^2r^2 \leq 0.$$

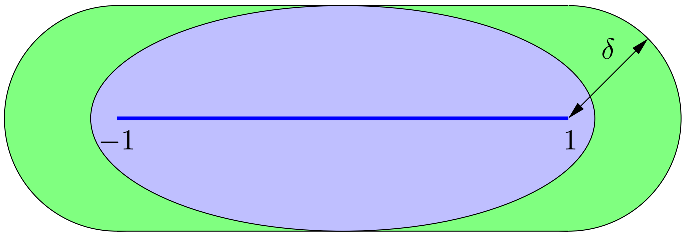

Figure 3.7: Bernstein disc included in a δ-neighbourhood of the interval $[-1,1]$

Due to $r = \|p\|_2 > 0$, the left equation implies either $b = 0$ or $ar + w_r = 0$.
In the first case, i.e., if $b = 0$, the right inequality takes the form

$$(ar + w_r)^2 + w_i^2 \le 0,$$

which implies $ar + w_r = 0$ and $w_i = 0$, i.e., $w_i = br$ and $|w_i| = |b|\,r$, i.e., $z \notin \mathcal{S}_w$.
In the second case, i.e., if $ar + w_r = 0$, the right inequality becomes

$$w_i^2 - b^2 r^2 \le 0 \iff w_i^2 \le b^2 r^2 \iff |w_i| \le |b|\,r.$$

We conclude again $z \notin \mathcal{S}_w$. ■

If w is known explicitly, we can find the major axis α of the Bernstein ellipse $\mathcal{E}_\varrho$ running through w via

$$\left|\frac{w}{r} - 1\right| + \left|\frac{w}{r} + 1\right| = 2\alpha$$

and compute ϱ by means of

$$\varrho + \frac{1}{\varrho} = 2\alpha \iff \varrho^2 - 2\alpha\varrho + 1 = 0 \iff (\varrho - \alpha)^2 = \alpha^2 - 1 \iff \varrho = \alpha \pm \sqrt{\alpha^2 - 1}.$$

In practice, we want to use an admissibility condition like (3.11), and this condition only provides us with an upper bound for

$$\begin{aligned}\frac{\gamma}{|w + \tau r|} &= \left|\frac{\gamma}{\sqrt{(w + \tau r)(\bar{w} + \tau r)}}\right| \\ &= |\hat{f}(\tau)| = \frac{\gamma}{\|m + \tau p\|_2} \le \frac{\gamma}{\operatorname{dist}(t,s)} \qquad \text{for all } \tau \in [-1,1],\end{aligned}$$

i.e., the distance between w and the interval $[-1,1]$ is at least $\operatorname{dist}(t,s)$, but we do not know exactly where w is positioned. Fortunately, if we stay close enough to $[-1,1]$, the triangle equality guarantees that we stay far enough from w, so we only have to find an Bernstein disc that is sufficiently close to $[-1,1]$, cf. Figure 3.7

Lemma 3.44 (Bernstein neighbourhood) *Let $\delta \in \mathbb{R}_{>0}$, and let $\varrho := \delta + \sqrt{\delta^2 + 1}$.*
For every $z \in \mathcal{D}_\varrho$, we can find $\tau \in [-1,1]$ with $|z - \tau| \le \delta$.

Proof. Let $z \in \mathcal{D}_\varrho$. Using Lemma 2.26, we find $w \in \mathbb{C}$ with $1 \leq |w| \leq \varrho$ and

$$z = \frac{w + 1/w}{2}.$$

We define $v := w/|w|$ and consider

$$\tau := \frac{v + 1/v}{2}.$$

Due to $|v| = 1$, we have

$$\tau = \frac{v + 1/v}{2} = \frac{v + \bar{v}/|v|^2}{2} = \frac{v + \bar{v}}{2} = \Re(v) \in \mathbb{R}, \qquad |\tau| \leq \frac{|v| + 1/|v|}{2} = 1,$$

and therefore $\tau \in [-1, 1]$. Using $|w| \geq 1$, we also find

$$\begin{aligned}
|z - \tau| &= \left| \frac{w + 1/w - v - 1/v}{2} \right| = \left| \frac{\left(1 - \frac{1}{|w|}\right) w + \frac{1-|w|}{w}}{2} \right| \\
&\leq \frac{\left|1 - \frac{1}{|w|}\right| |w| + \frac{|1-|w||}{|w|}}{2} = \frac{|w| - 1 + \frac{|w|-1}{|w|}}{2} = \frac{|w| - 1/|w|}{2} \\
&\leq \frac{\varrho - 1/\varrho}{2} = \frac{(\delta + \sqrt{\delta^2+1})^2 - 1}{2(\delta + \sqrt{\delta^2+1})} = \frac{\delta^2 + 2\delta\sqrt{\delta^2+1} + \delta^2 + 1 - 1}{2(\delta + \sqrt{\delta^2+1})} = \delta.
\end{aligned}$$

■

In order to apply Theorem 2.29, we need an upper bound for $\hat{f}$ in a Bernstein disc that is as large as possible.

Lemma 3.45 (Upper bound) *Let $\delta_{max} := \frac{2\,\mathrm{dist}(t,s)}{\mathrm{diam}(t)}$, and let $\delta \in [0 : \delta_{max})$. Let $\varrho := \delta + \sqrt{\delta^2 + 1}$. We have*

$$|\hat{f}(z)| \leq \frac{\delta_{max}}{\delta_{max} - \delta} \frac{\gamma}{\mathrm{dist}(t,s)} \qquad \text{for all } z \in \mathcal{D}_\varrho.$$

Proof. Let $z \in \mathcal{D}_\varrho$. Using Lemma 3.44, we find $\tau \in [-1, 1]$ with $|z - \tau| \leq \delta$.

Using the triangle inequality, we obtain

$$\begin{aligned}
|w + zr| &\geq |w + \tau r| - |\tau - z| r \geq \|m + \tau p\|_2 - \delta r \geq \mathrm{dist}(t,s) - \delta \frac{\mathrm{diam}(t)}{2} \\
&= \mathrm{dist}(t,s) - \frac{\delta}{\delta_{\max}} \delta_{\max} \frac{\mathrm{diam}(t)}{2} = \left(1 - \frac{\delta}{\delta_{\max}}\right) \mathrm{dist}(t,s), \\
|\bar{w} + zr| &\geq |\bar{w} + \tau r| - |\tau - z| r = |w + \tau r| - |\tau - z| r \geq \left(1 - \frac{\delta}{\delta_{\max}}\right) \mathrm{dist}(t,s), \\
|\hat{f}(z)| &= \frac{\gamma}{|\sqrt{(w + zr)(\bar{w} + zr)}|} = \frac{\gamma}{\sqrt{|w + zr|\,|\bar{w} + zr|}} \\
&\leq \frac{\gamma}{(1 - \delta/\delta_{\max})\,\mathrm{dist}(t,s)}.
\end{aligned}$$

■

Theorem 3.46 (Partial approximation) *Let* $\delta_{max} := \frac{2\operatorname{dist}(t,s)}{\operatorname{diam}(t)} > 0$ *and* $\varrho_{max} := \delta_{max} + \sqrt{\delta_{max}^2 + 1}$. *For every* $m \in \mathbb{N}$, *we can find a polynomial* $p \in \Pi_m$ *with*

$$\|\hat{f} - p\|_{\infty,[-1,1]} \leq \frac{2e(m+1)^2}{m\,\delta_{max}} \frac{\gamma}{\operatorname{dist}(t,s)} \varrho_{max}^{-m}.$$

Proof. We define

$$\delta := \frac{m}{m+1}\delta_{\max}, \qquad\qquad \varrho := \delta + \sqrt{\delta^2+1}.$$

Due to $\delta_{\max} > 0$, we have $\delta > 0$ and $\varrho > 1$. Lemma 3.45 yields

$$|\hat{f}(z)| \leq \frac{\delta_{\max}}{\delta_{\max} - \delta} \frac{\gamma}{\operatorname{dist}(t,s)} = \frac{\gamma(m+1)}{\operatorname{dist}(t,s)} \qquad\qquad \text{for all } z \in \mathcal{D}_\varrho,$$

and we can apply Theorem 2.29 with $\hat{\varrho} = 1$ to find a polynomial $p \in \Pi_m$ with

$$\|\hat{f} - p\|_{\infty,[-1,1]} \leq \frac{2}{\varrho - 1}\left(\frac{1}{\varrho}\right)^m \frac{\gamma(m+1)}{\operatorname{dist}(t,s)}.$$

We have

$$\begin{aligned}
\varrho = \delta + \sqrt{\delta^2+1} &= \frac{m}{m+1}\delta_{\max} + \sqrt{\left(\frac{m}{m+1}\right)^2 \delta_{\max}^2 + 1} \\
&\geq \frac{m}{m+1}\delta_{\max} + \sqrt{\left(\frac{m}{m+1}\right)^2 (\delta_{\max}^2 + 1)} \\
&= \frac{m}{m+1}\left(\delta_{\max} + \sqrt{\delta_{\max}^2+1}\right) = \frac{m}{m+1}\varrho_{\max}, \\
\varrho - 1 = \delta + \sqrt{\delta^2+1} - 1 &\geq \delta = \frac{m}{m+1}\delta_{\max},
\end{aligned}$$

and conclude

$$\begin{aligned}
\|\hat{f} - p\|_{\infty,[-1,1]} &\leq \frac{2}{\varrho-1}\left(\frac{1}{\varrho}\right)^m \frac{\gamma\,(m+1)}{\operatorname{dist}(t,s)} \leq \frac{2}{\frac{m}{m+1}\delta_{\max}}\left(\frac{m+1}{m}\right)^m \frac{\gamma\,(m+1)}{\operatorname{dist}(t,s)}\varrho_{\max}^{-m} \\
&\leq \frac{2(m+1)^2}{m\,\delta_{\max}}\left(1 + \frac{1}{m}\right)^m \frac{\gamma}{\operatorname{dist}(t,s)}\varrho_{\max}^{-m} \leq \frac{2e(m+1)^2}{m\,\delta_{\max}} \frac{\gamma}{\operatorname{dist}(t,s)}\varrho_{\max}^{-m}
\end{aligned}$$

∎

Corollary 3.47 (Kernel function) *Let* $\operatorname{diam}(t) \leq 2\eta \operatorname{dist}(t,s)$. *Let* $m \in \mathbb{N}$. *Let* Λ_m *be the stability constant of (2.24). We have*

$$\|g - \tilde{g}_{ts}\|_{\infty,t\times s} \leq 2de\Lambda_m^{d-1}(1+\Lambda_m)\eta \frac{(m+1)^2}{m} \frac{\gamma}{\operatorname{dist}(t,s)} \left(\frac{\eta}{1+\sqrt{\eta^2+1}}\right)^m.$$

Proof. Let $x \in t$ and $y \in s$. Theorem 3.6 yields

$$|g(x,y) - \tilde{g}_{ts}(x,y)| = |f(x) - \mathfrak{I}_t[f](x)| \leq \sum_{\iota=1}^{d} \Lambda_m^{\iota-1} \|f - \mathfrak{I}_{t,\iota}[f]\|_{\infty,t}$$

for the function f defined in Lemma 3.41. For $\iota \in [1:d]$, this lemma states

$$|f(x) - \mathfrak{I}[f](x)| \leq (1 + \Lambda_m) \|\hat{f}_\iota - p\|_{\infty,[-1,1]} \qquad \text{for all } p \in \Pi_m.$$

Finally, Theorem 3.46 provides us with a polynomial $p \in \Pi_m$ such that

$$\|\hat{f}_\iota - p\|_{\infty,[-1,1]} \leq \frac{2e(m+1)^2}{m\delta_{\max}} \frac{\gamma}{\operatorname{dist}(t,s)} \varrho_{\max}^{-m}$$

with

$$\delta_{\max} = \frac{2\operatorname{dist}(t,s)}{\operatorname{diam}(t)} \geq \frac{1}{\eta}, \qquad \varrho_{\max} = \delta_{\max} + \sqrt{\delta_{\max}^2 + 1} \geq \frac{1 + \sqrt{\eta^2 + 1}}{\eta}.$$

Combining these estimates yields the result. ∎

4 Low-rank matrices

Most approximation schemes for non-local operators make use of low-rank matrices to express long-range interactions by a small number of coefficients. In this chapter, we consider techniques for obtaining low-rank approximations.

4.1 Definition and basic properties

In order to approximate an arbitrary matrix by an $\mathcal{H}$-matrix, we have to find low-rank approximations of all admissible submatrices.

Let $\mathcal{I}$ and $\mathcal{J}$ be finite index sets, and let $n_{\mathcal{I}} := |\mathcal{I}|$ and $n_{\mathcal{J}} := |\mathcal{J}|$ denote their cardinalities.

For convenience, we introduce the following notations.

Definition 4.1 (Range) *Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$. The set*

$$\operatorname{range}(X) := \{Xy \ : \ y \in \mathbb{K}^{\mathcal{J}}\}$$

is called the range *of X. It is a subspace of $\mathbb{K}^{\mathcal{I}}$.*

Definition 4.2 (Rank) *Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$. The* rank *of X is given by*

$$\operatorname{rank}(X) := \dim \operatorname{range}(X).$$

Due to $\operatorname{range}(X) \subseteq \mathbb{K}^{\mathcal{I}}$*, the rank is bounded by $n_{\mathcal{I}}$.*

Notation 4.3 (Matrices, vectors, scalars) *We identify $\mathbb{K}^{\mathcal{I}}$ and $\mathbb{K}^{\mathcal{I}\times 1}$, i.e., each vector can also be treated as a matrix with only one column containing its entries.*

We also identify $\mathbb{K}$ and $\mathbb{K}^{1\times 1}$, i.e., a one-by-one matrix can be treated as a scalar.

Most importantly, these notations imply that x^*y is the Euclidean inner product of $x, y \in \mathbb{K}^{\mathcal{I}}$, and that $\|x\|_2 := \sqrt{x^*x}$ is the Euclidean norm of $x \in \mathbb{K}^{\mathcal{I}}$.

Using these notations, we can construct matrices of rank one in factorized form: let $a \in \mathbb{K}^{\mathcal{I}} \setminus \{0\}$ and $b \in \mathbb{K}^{\mathcal{J}} \setminus \{0\}$. Then

$$R := ab^* \tag{4.1}$$

is a matrix in $\mathbb{K}^{\mathcal{I}\times\mathcal{J}}$. Its range is contained in the span of a and due to $Rb = ab^*b = a\|b\|_2^2 \neq 0$ it is also non-trivial, so R is a rank-one matrix.

We can prove that *any* rank-one matrix can be represented in this way.

Lemma 4.4 (Rank reduction) *Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ be a matrix of rank k, and let $a \in \operatorname{range}(X)$ and $c \in \mathbb{K}^{\mathcal{I}}$ be vectors with $c^*a = 1$. Then*

$$\widetilde{X} := X - ac^*X$$

is a matrix of rank $k-1$.

Proof. Due to $a \neq 0$, $a \in \operatorname{range}(X)$ and $\operatorname{rank}(X) = \dim\operatorname{range}(X) = k$, we can apply the basis extension theorem to find $b_1, \ldots, b_{k-1} \in \mathbb{K}^{\mathcal{I}}$ such that $\{a, b_1, \ldots, b_{k-1}\}$ is a basis of $\operatorname{range}(X)$.

We define

$$d_i := b_i - ac^*b_i \qquad \text{for all } i \in [1 : k-1]$$

and will prove that $\{d_1, \ldots, d_{k-1}\}$ is a basis of $\operatorname{range}(\widetilde{X})$.

We first prove that $\operatorname{range}(\widetilde{X})$ is contained in the span of $d_1, \ldots, d_{k-1}$. Let $y \in \mathbb{K}^{\mathcal{J}}$. Due to $Xy \in \operatorname{range}(X)$, we can find $\alpha, \beta_1, \ldots, \beta_{k-1} \in \mathbb{K}$ such that

$$Xy = \alpha a + \beta_1 b_1 + \ldots + \beta_{k-1} b_{k-1}.$$

By our definitions, we have

$$\begin{aligned}
\widetilde{X}y &= Xy - ac^*Xy \\
&= \alpha a + \beta_1 b_1 + \ldots + \beta_{k-1}b_{k-1} - ac^*(\alpha a + \beta_1 b_1 + \ldots + \beta_{k-1}b_{k-1}) \\
&= \alpha a + \beta_1 b_1 + \ldots + \beta_{k-1}b_{k-1} - \alpha ac^*a - \beta_1 ac^*b_1 - \ldots - \beta_{k-1}ac^*b_{k-1} \\
&= \alpha(1 - c^*a)a + \beta_1(b_1 - ac^*b_1) + \ldots + \beta_{k-1}(b_{k-1} - ac^*b_{k-1}) \\
&= \alpha(1 - c^*a)a + \beta_1 d_1 + \ldots + \beta_{k-1}d_{k-1}.
\end{aligned}$$

Due to $c^*a = 1$, the first term vanishes and we have proven our claim.

Now we will prove that the vectors $d_1, \ldots, d_{k-1}$ are linearly independent. Let $\gamma_1, \ldots, \gamma_{k-1} \in \mathbb{K}$ be such that

$$\gamma_1 d_1 + \ldots + \gamma_{k-1}d_{k-1} = 0.$$

Our definition implies

$$\begin{aligned}
0 &= \gamma_1 d_1 + \ldots + \gamma_{k-1}d_{k-1} \\
&= \gamma_1(b_1 - ac^*b_1) + \ldots + \gamma_{k-1}(b_{k-1} - ac^*b_{k-1}) \\
&= \gamma_1 b_1 + \ldots + \gamma_{k-1}b_{k-1} - (\gamma_1 c^*b_1 + \ldots + \gamma_{k-1}c^*b_{k-1})a.
\end{aligned}$$

Since the vectors $\{a, b_1, \ldots, b_{k-1}\}$ are linearly independent, we conclude $\gamma_1 = \ldots = \gamma_{k-1} = 0$. This completes our proof. ∎

We apply Lemma 4.4 to a rank-one matrix R: we choose $a \in \operatorname{range}(R)$ with $\|a\|_2 = 1$, let $c := a$ and find that $\widetilde{R} := R - aa^*R$ is a matrix of rank zero. This means that its range is trivial, so we have $\widetilde{R} = 0$ and conclude $R = aa^*R = a(R^*a)^*$. Setting $b := R^*a$, we have proven $R = ab^*$.

Remark 4.5 (Projection) *Due to $c^*a = 1$, the mapping $P := I - ac^*$ appearing in Lemma 4.4 is a projection, i.e., satisfies $P^2 = P$, and its nullspace is the span of a.*

This means that P is injective on complements of a, e.g., on the space spanned by $b_1, \dots, b_{k-1}$. Therefore the vectors $d_i = Pb_i$ are linear independent and span the range of $\widetilde{X} = PX$.

We can extend this result: matrices of rank k can be represented as sums of k rank-one matrices, each of which can again be represented in factorized form.

Theorem 4.6 (Low-rank representation) *Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ be a matrix of rank k. We can find vectors $a_1, \dots, a_k \in \mathbb{K}^{\mathcal{I}}$ and $b_1, \dots, b_k \in \mathbb{K}^{\mathcal{J}}$ such that*

$$X = \sum_{\nu=1}^{k} a_\nu b_\nu^*. \tag{4.2}$$

Proof. By induction.

Let first $k = 1$, and let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ be a matrix of rank one. We can find $a \in \operatorname{range}(X)$ with $\|a\|_2 = 1$, and applying Lemma 4.4 to $c := a$ with $c^*a = \|a\|_2^2 = 1$ yields that $X - aa^*X$ is a matrix of rank zero, i.e.,

$$X = aa^*X = a(X^*a)^* = ab^*,$$

where we have chosen $b := X^*a$.

Let now $k \in \mathbb{N}$ be given such that our claim holds. Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ be a matrix of rank $k+1$. We can again find $a \in \operatorname{range}(X)$ with $\|a\|_2 = 1$, and Lemma 4.4 again yields that

$$\widetilde{X} := X - aa^*X$$

is a matrix of rank k. Applying the assumption yields vectors $a_1, \dots, a_k \in \mathbb{K}^{\mathcal{I}}$ and $b_1, \dots, b_k \in \mathbb{K}^{\mathcal{J}}$ such that

$$\widetilde{X} = \sum_{\nu=1}^{k} a_\nu b_\nu^*,$$

and setting $a_{k+1} := a$ and $b_{k+1} := X^*a$, we conclude

$$X = \widetilde{X} + aa^*X = \sum_{\nu=1}^{k} a_\nu b_\nu^* + a_{k+1}b_{k+1}^* = \sum_{\nu=1}^{k+1} a_\nu b_\nu^*.$$

■

In order to avoid sums and indices for this representation of low-rank matrices, we collect the vectors $a_1, \dots, a_k$ in the columns of a matrix A and the vectors $b_1, \dots, b_k$ in the columns of a matrix B and obtain a compact notation that naturally extends (4.1) to the case of higher ranks.

Since A and B now have column indices in $[1 : k]$ and row indices in $\mathcal{I}$ and $\mathcal{J}$, we introduce a suitable notation.

Notation 4.7 (Column matrices) *When dealing with matrices with column indices in $\mathbb{N}$, we write $\mathbb{K}^{\mathcal{I}\times k} := \mathbb{K}^{\mathcal{I}\times[1:k]}$ and $\mathbb{K}^{k\times\mathcal{J}} := \mathbb{K}^{[1:k]\times\mathcal{J}}$.*

Correspondingly, for $A \in \mathbb{K}^{\mathcal{I}\times k}$ and $\ell \in [1:k]$, we define the restriction to the first ℓ columns as $A|_{\hat{t}\times\ell} := A|_{\hat{t}\times[1:\ell]}$ for subsets $\hat{t} \subseteq \mathcal{I}$.

Definition 4.8 ($\mathcal{R}(\mathcal{I},\mathcal{J},k)$-representation) *Let $k \in \mathbb{N}$ and $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$. We call a pair (A,B) of matrices $A \in \mathbb{K}^{\mathcal{I}\times k}$ and $B \in \mathbb{K}^{\mathcal{J}\times k}$ an $\mathcal{R}(\mathcal{I},\mathcal{J},k)$-*representation *of X if*

$$X = AB^* \tag{4.3}$$

and we denote the set of all these representations by

$$\mathcal{R}(\mathcal{I},\mathcal{J},k) := \{AB^* \ :\ A \in \mathbb{K}^{\mathcal{I}\times k},\ B \in \mathbb{K}^{\mathcal{J}\times k}\}.$$

This is not a vector space, since the sum of two rank-k-matrices can have a rank higher than k. If the index sets $\mathcal{I}$ and $\mathcal{J}$ are irrelevant or implied by the context, we use the short notation $\mathcal{R}(k)$.

Using this notation, Theorem 4.6 takes the form of the following existence result for $\mathcal{R}(\mathcal{I},\mathcal{J},k)$-representations.

Corollary 4.9 ($\mathcal{R}(\mathcal{I},\mathcal{J},k)$-representation) *Let $k \in \mathbb{N}$ and $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$.*

If X is of rank k, we have $X \in \mathcal{R}(\mathcal{I},\mathcal{J},k)$.

If $X \in \mathcal{R}(\mathcal{I},\mathcal{J},k)$, we have $\operatorname{rank}(X) \leq k$.

Proof. Let first X be of rank k. Applying Theorem 4.6, we find vectors $a_1,\ldots,a_k \in \mathbb{K}^{\mathcal{I}}$ and $b_1,\ldots,b_k \in \mathbb{K}^{\mathcal{J}}$ such that

$$X = \sum_{\nu=1}^{k} a_\nu b_\nu^*.$$

We define

$$A := \begin{pmatrix} a_1 & \ldots & a_k \end{pmatrix} \in \mathbb{K}^{\mathcal{I}\times k}, \qquad B := \begin{pmatrix} b_1 & \ldots & b_k \end{pmatrix} \in \mathbb{K}^{\mathcal{J}\times k}$$

and obtain $X = AB^*$, i.e., $X \in \mathcal{R}(\mathcal{I},\mathcal{J},k)$.

Let now $X \in \mathcal{R}(\mathcal{I},\mathcal{J},k)$. By definition, we find $A \in \mathbb{K}^{\mathcal{I}\times k}$ and $B \in \mathbb{K}^{\mathcal{J}\times k}$ with $X = AB^*$. We have $\operatorname{range}(X) \subseteq \operatorname{range}(A)$, and since A has only k columns, we conclude

$$\dim \operatorname{range}(X) \leq \dim \operatorname{range}(A) \leq k,$$

i.e., $\operatorname{rank}(X) \leq k$. ■

Remark 4.10 (Storage) *A matrix in $\mathcal{R}(\mathcal{I},\mathcal{J},k)$-representation requires $(n_{\mathcal{I}} + n_{\mathcal{J}})k$ units of storage. If the rank is low compared to the number $n_{\mathcal{I}}$ of rows and the number $n_{\mathcal{J}}$ of columns, this representation is far more efficient than the representation as a standard two-dimensional $n_{\mathcal{I}}$-by-$n_{\mathcal{J}}$ array.*

Remark 4.11 (Matrix-vector multiplication) *Computing the product of a matrix X in $\mathcal{R}(\mathcal{I},\mathcal{J},k)$-representation and a vector $y \in \mathbb{K}^{\mathcal{J}}$ takes $2(n_{\mathcal{I}}+n_{\mathcal{J}})k$ operations: let $X = AB^*$ with $A \in \mathbb{K}^{\mathcal{I}\times k}$ and $B \in \mathbb{K}^{\mathcal{J}\times k}$. In order to compute $z := Xy$, we first compute the auxiliary vector $\hat{y} := B^*y$, using not more than $2n_{\mathcal{J}}k$ operations, and then $z = A\hat{y}$, using not more than $2n_{\mathcal{I}}k$ operations.*

Using the representation (4.2), we can carry out this operation with minimal auxiliary storage: we start by initializing $z \leftarrow 0$. Then we perform the update $z \leftarrow z + a_\nu b_\nu^* y$ for all $\nu \in [1:k]$ by first computing the inner product $\hat{y}_\nu := b_\nu^* y$ and then updating $z \leftarrow z + a_\nu \hat{y}_\nu$.

Remark 4.12 (Matrix-matrix multiplication) *If $X \in \mathcal{R}(\mathcal{I},\mathcal{J},k)$ and $Y \in \mathbb{K}^{\mathcal{J}\times\mathcal{K}}$, we have $XY \in \mathcal{R}(\mathcal{I},\mathcal{K},k)$: using $X = AB^*$, we obtain $XY = AB^*Y = A(Y^*B)^*$, and setting $C := Y^*B$, we have found the $\mathcal{R}(\mathcal{I},\mathcal{K},k)$-representation $XY = AC^*$.*

Similarly, if $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ and $Y \in \mathcal{R}(\mathcal{J},\mathcal{K},k)$, we also have $XY \in \mathcal{R}(\mathcal{I},\mathcal{K},k)$.

Exercise 4.13 (Uniqueness) *So far, we have only considered the existence of low-rank representations. Sometimes the uniqueness may also be of interest.*

(a) Let $a_1, a_2 \in \mathbb{K}^{\mathcal{I}}$ and $b_1, b_2 \in \mathbb{K}^{\mathcal{J}}$.

Prove that $a_1 b_1^* = a_2 b_2^* \neq 0$ implies that there is a constant $\gamma \in \mathbb{K}\setminus\{0\}$ such that $a_2 = \gamma a_1$ and $b_2 = \frac{1}{\gamma} b_1$.

(b) Let $A_1, A_2 \in \mathbb{K}^{\mathcal{I}\times k}$ and $B_1, B_2 \in \mathbb{K}^{\mathcal{J}\times k}$.

Prove that if $A_1 B_1^* = A_2 B_2^*$ is a matrix of rank k, there is an invertible matrix C such that $A_2 = A_1 C$ and $B_2 = B_1 (C^{-1})^*$.

Exercise 4.14 (Self-adjoint matrices) *Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{I}}$ be a self-adjoint matrix.*

(a) Assume that X is positive semidefinite*, i.e., that*

$$y^*Xy \in \mathbb{R}_{\geq 0} \qquad \text{for all } y \in \mathbb{K}^{\mathcal{I}}.$$

Prove that $y^*Xy = 0$ implies $Xy = 0$ for all $y \in \mathbb{K}^{\mathcal{I}}$.

(b) Since the unit sphere in $\mathbb{K}^{\mathcal{I}}$ is compact (cf. the Heine-Borel theorem), there is a vector $e \in \mathbb{K}^{\mathcal{I}}$ with $e^*e = \|e\|_2^2 = 1$ and

$$y^*Xy \leq e^*Xe \qquad \text{for all } y \in \mathbb{K}^{\mathcal{I}}.$$

Prove that there is a $\lambda \in \mathbb{R}$ with $Xe = \lambda e$.

(c) Let $k := \operatorname{rank}(X)$. Prove that there are $\lambda_1, \ldots, \lambda_k \in \mathbb{R}$ and $e_1, \ldots, e_k \in \mathbb{K}^{\mathcal{I}}$ with

$$X = \sum_{\nu=1}^{k} e_\nu \lambda_\nu e_\nu^*.$$

Hints: *Considering $(y + \alpha Xy)^* X (y + \alpha Xy)$ for $\alpha \to 0$ may be useful in part (a). For part (b), it may be a good idea to look for a $\lambda \in \mathbb{R}$ such that the matrix $\lambda I - X$ is positive semidefinite. Part (c) can be solved by combining part (b) and Lemma 4.4.*

4.2 Low-rank approximation

In most applications, we can only hope to *approximate* matrices by low-rank matrices. Therefore it would be particularly attractive to have a reliable algorithm for finding the *best* approximation of a given matrix by a matrix of given rank k. We will now introduce this algorithm.

Definition 4.15 (Orthogonal and isometric matrices) *Let $Q \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$. If we have $Q^*Q = I$, we call Q an* isometric *matrix.*
If we have $Q^*Q = I$ and $QQ^* = I$, we call Q an orthogonal *matrix.*

If $Q \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ is isometric, we have

$$(Qx)^*(Qy) = x^*Q^*Qy = x^*y \qquad \text{for all } x, y \in \mathbb{K}^{\mathcal{J}}, \tag{4.4a}$$

$$\|Qx\|_2 = \sqrt{(Qx)^*(Qx)} = \sqrt{x^*x} = \|x\|_2 \qquad \text{for all } x \in \mathbb{K}^{\mathcal{J}}. \tag{4.4b}$$

The latter equation motivates the name "isometric". It is possible to prove that it is equivalent to the equation $Q^*Q = I$ used in the definition.

If Q is orthogonal, it is both injective and surjective, and is therefore an isometric isomorphism between $\mathbb{K}^{\mathcal{I}}$ and $\mathbb{K}^{\mathcal{J}}$.

Definition 4.16 (Singular value decomposition) *Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$, and let $p :=$ rank(X). If there are isometric matrices $U \in \mathbb{K}^{\mathcal{I}\times p}$ and $V \in \mathbb{K}^{\mathcal{J}\times p}$ and a diagonal matrix*

$$\Sigma := \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_p \end{pmatrix} \in \mathbb{R}^{p\times p}$$

with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p > 0$ such that

$$X = U\Sigma V^*, \tag{4.5}$$

we call (U, Σ, V) a singular value decomposition *of X.*

We can rewrite (4.5) as

$$X = \sum_{\nu=1}^{p} u_\nu \sigma_\nu v_\nu^*$$

to make it resemble (4.2), where u_ν and v_ν are the ν-th columns of U and V, respectively. The only difference compared to the result of Theorem 4.6 lies in the fact that $\{u_1, \ldots, u_p\}$ and $\{v_1, \ldots, v_p\}$ are now *orthonormal* families of vectors. This property makes the singular value decomposition significantly more useful, but it also makes it significantly harder to obtain.

Reminder 4.17 (Cauchy-Schwarz inequality) *For all $x, y \in \mathbb{K}^{\mathcal{I}}$, we have*

$$|x^*y| \leq \|x\|_2\|y\|_2.$$

Equality holds only if x and y are linear dependent.

Theorem 4.18 (Singular value decomposition) *Every matrix $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ has a singular value decomposition.*

Proof. By induction on $\operatorname{rank}(X)$.

For $\operatorname{rank}(X) = 0$, we choose empty matrices U, V and Σ and are done.

Let now $p \in \mathbb{N}_0$ be such that every matrix $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ with $\operatorname{rank}(X) = p$ has a singular value decomposition.

Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ be a matrix with $\operatorname{rank}(X) = p + 1$. We consider the function

$$f : \mathbb{K}^{\mathcal{I}} \times \mathbb{K}^{\mathcal{J}} \to \mathbb{R}_{\geq 0}, \qquad (z, y) \mapsto |z^* X y|,$$

on the unit spheres

$$S_{\mathcal{I}} := \{z \in \mathbb{K}^{\mathcal{I}} \ : \ \|z\|_2 = 1\}, \qquad S_{\mathcal{J}} := \{y \in \mathbb{K}^{\mathcal{J}} \ : \ \|y\|_2 = 1\}.$$

Since the unit spheres are compact and f is continuous, we can find a maximum

$$\sigma := \max\{f(z, y) \ : \ z \in S_{\mathcal{I}},\ y \in S_{\mathcal{J}}\}.$$

Due to $\operatorname{rank}(X) = p+1 > 0$, we have $X \neq 0$ and therefore $\sigma > 0$. Let $u \in S_{\mathcal{I}}$ and $v \in S_{\mathcal{J}}$ be vectors with $f(u, v) = \sigma$. By modifying the sign of u, we can ensure $\sigma = u^* X v$.

We will now prove $Xv = \sigma u$ and $X^* u = \sigma v$.

In order to prove $Xv = \sigma u$, we make use of $Xv \neq 0$ to define

$$z := \frac{Xv}{\|Xv\|_2}.$$

Due to $z \in S_{\mathcal{I}}$, we have

$$|u^* X v| = \sigma \geq f(z, v) = \frac{|(Xv)^* X v|}{\|Xv\|_2} = \frac{\|Xv\|_2^2}{\|Xv\|_2} = \|Xv\|_2 = \|u\|_2 \|Xv\|_2.$$

By the Cauchy-Schwarz lemma, $|u^* X v| \geq \|u\|_2 \|Xv\|_2$ can only hold if u and Xv are linearly dependent, so we find $\lambda \in \mathbb{K}$ with $Xv = \lambda u$.

$$\sigma = u^* X v = \lambda u^* u = \lambda \|u\|_2^2 = \lambda$$

yields the desired equation $Xv = \sigma u$.

In order to prove $X^* u = \sigma v$, we make use of $X^* u \neq 0$ to define

$$y := \frac{X^* u}{\|X^* u\|_2}$$

with $y \in S_{\mathcal{J}}$. Proceeding as before, we find $|v^* X^* u| \geq |y^* X^* u| = \|v\|_2 \|X^* u\|_2$ and finally $X^* u = \sigma v$.

We consider the matrix

$$\widetilde{X} := X - u u^* X.$$

Due to $Xv = \sigma u$ and $\sigma \neq 0$, we have $u \in \operatorname{range}(X)$, and since we also have $u^*u = \|u\|_2^2 = 1$, we can apply Lemma 4.4 to conclude $\operatorname{rank}(\widetilde{X}) = p$.

Using the induction assumption yields a singular value decomposition $(\widetilde{U}, \widetilde{\Sigma}, \widetilde{V})$ of $\widetilde{X}$.

We define

$$U := \begin{pmatrix} u & \widetilde{U} \end{pmatrix}, \qquad \Sigma := \begin{pmatrix} \sigma & \\ & \widetilde{\Sigma} \end{pmatrix}, \qquad V := \begin{pmatrix} v & \widetilde{V} \end{pmatrix}$$

and will now verify that (U, Σ, V) is a singular value decomposition of X.

First we prove that U and V are isometric. We have

$$U^*U = \begin{pmatrix} u & \widetilde{U} \end{pmatrix}^* \begin{pmatrix} u & \widetilde{U} \end{pmatrix} = \begin{pmatrix} u^* \\ \widetilde{U}^* \end{pmatrix} \begin{pmatrix} u & \widetilde{U} \end{pmatrix} = \begin{pmatrix} u^*u & u^*\widetilde{U} \\ \widetilde{U}^*u & \widetilde{U}^*\widetilde{U} \end{pmatrix} = \begin{pmatrix} 1 & u^*\widetilde{U} \\ \widetilde{U}^*u & I \end{pmatrix},$$
$$V^*V = \begin{pmatrix} v & \widetilde{V} \end{pmatrix}^* \begin{pmatrix} v & \widetilde{V} \end{pmatrix} = \begin{pmatrix} v^* \\ \widetilde{V}^* \end{pmatrix} \begin{pmatrix} v & \widetilde{V} \end{pmatrix} = \begin{pmatrix} v^*v & v^*\widetilde{V} \\ \widetilde{V}^*v & \widetilde{V}^*\widetilde{V} \end{pmatrix} = \begin{pmatrix} 1 & v^*\widetilde{V} \\ \widetilde{V}^*v & I \end{pmatrix}.$$

We only have to show $u^*\widetilde{U} = 0$ and $\widetilde{V}^*v = 0$. Since $\widetilde{\Sigma}$ is invertible, we have

$$\widetilde{X}\widetilde{V}\widetilde{\Sigma}^{-1} = \widetilde{U}\widetilde{\Sigma}\widetilde{V}^*\widetilde{V}\widetilde{\Sigma}^{-1} = \widetilde{U}\widetilde{\Sigma}\widetilde{\Sigma}^{-1} = \widetilde{U},$$
$$\widetilde{\Sigma}^{-1}\widetilde{U}^*\widetilde{X} = \widetilde{\Sigma}^{-1}\widetilde{U}^*\widetilde{U}\widetilde{\Sigma}\widetilde{V}^* = \widetilde{\Sigma}^{-1}\widetilde{\Sigma}\widetilde{V}^* = \widetilde{V}^*.$$

This implies

$$u^*\widetilde{U} = u^*\widetilde{X}\widetilde{V}\widetilde{\Sigma}^{-1} = u^*(X - uu^*X)\widetilde{V}\widetilde{\Sigma}^{-1} = (u^*X - u^*X)\widetilde{V}\widetilde{\Sigma}^{-1} = 0, \tag{4.6}$$

i.e., $U^*U = I$. In order to prove $V^*V = I$, we observe

$$\widetilde{X} = X - uu^*X = X - u(X^*u)^* = X - \sigma uv^* = X - Xvv^*$$

and can now use the same argument as before to get

$$\widetilde{V}^*v = \widetilde{\Sigma}^{-1}\widetilde{U}^*\widetilde{X}v = \widetilde{\Sigma}^{-1}\widetilde{U}^*(X - Xvv^*)v = \widetilde{\Sigma}^{-1}\widetilde{U}^*(Xv - Xv) = 0,$$

i.e., $V^*V = I$. To complete the proof, we have to show that σ is larger than all diagonal entries of $\widetilde{\Sigma}$. Let $\nu \in [1:p]$, and let $\tilde{\sigma}_\nu$ denote the ν-th diagonal element of $\widetilde{\Sigma}$.

We denote the ν-th columns of $\widetilde{U}$ and $\widetilde{V}$ by $\tilde{u} \in \mathbb{K}^{\mathcal{I}}$ and $\tilde{v} \in \mathbb{K}^{\mathcal{J}}$. Since $\widetilde{U}$ and $\widetilde{V}$ are isometric, we have $\tilde{u} \in S_{\mathcal{I}}$ and $\tilde{v} \in S_{\mathcal{J}}$. Equation (4.6) implies $u^*\tilde{u} = 0$, so we find

$$\sigma \geq f(\tilde{u}, \tilde{v}) = |\tilde{u}^*X\tilde{v}| = |\tilde{u}^*(X - uu^*X)\tilde{v}| = |\tilde{u}^*\widetilde{X}\tilde{v}| = |\tilde{u}^*\widetilde{U}\widetilde{\Sigma}\widetilde{V}^*\tilde{v}| = \tilde{\sigma}_\nu,$$

since both $\widetilde{U}^*\tilde{u}$ and $\widetilde{V}^*\tilde{v}$ are the ν-th canonical unit vector. ■

Lemma 4.19 (Range and null space) *Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$, let $p := \operatorname{rank}(X)$, and let (U, Σ, V) be a singular value decomposition of X.*

We have $\operatorname{range}(X) = \operatorname{range}(U)$ *and* $\operatorname{range}(X^*) = \operatorname{range}(V)$.

For the null spaces we have $\ker(X) = \ker(V^*)$ *and* $\ker(X^*) = \ker(U^*)$.

The dimensions satisfy

$$\dim\operatorname{range}(X) = \dim\operatorname{range}(U) = p, \qquad \dim\ker(X) = \dim\ker(V^*) = |\mathcal{J}| - p.$$

Proof. The defining equation $X = U\Sigma V^*$ already implies range$(X) \subseteq$ range(U). Since both range(X) and range(U) are p-dimensional, we conclude range$(X) =$ range(U).

Applying the same argument to $X^* = V\Sigma U^*$, we obtain range$(X^*) =$ range(V).

The defining equation also yields $\ker(V^*) \subseteq \ker(X)$. Let $y \in \ker(X)$. We have

$$0 = \|Xy\|_2 = \|U\Sigma V^* y\|_2 = \|\Sigma V^* y\|_2$$

due to (4.4b), i.e., $\Sigma V^* y = 0$. Since Σ is invertible, this implies $V^* y = 0$, i.e., $y \in \ker(V^*)$. The same reasoning leads to $\ker(U^*) = \ker(X^*)$.

The rank-nullity theorem yields

$$|\mathcal{J}| = \dim \operatorname{range}(X) + \dim \ker(X) = \dim \operatorname{range}(U) + \dim \ker(V^*),$$

and $\dim \operatorname{range}(X) = p = \dim \operatorname{range}(U)$ completes the proof. ■

If the rank p is too large for our purposes, we can reduce it to a lower value $k \leq p$ by dropping all but the first k terms of the representation, i.e., we can consider the approximation

$$\widetilde{X} := \sum_{\nu=1}^{k} u_\nu \sigma_\nu v_\nu^* = U \begin{pmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_k & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix} V^*. \tag{4.7}$$

Obviously it would be of great interest to know how good this approximation actually is. In order to measure the approximation error, we introduce appropriate norms.

Definition 4.20 (Frobenius norm) *The* Frobenius norm *of a matrix $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ is given by*

$$\|X\|_F := \left(\sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} |x_{ij}|^2 \right)^{1/2}.$$

Lemma 4.21 (Frobenius norm) *Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$. We have*

$$\|X\|_F = \|X^*\|_F, \tag{4.8a}$$

$$\|Xy\|_2 \leq \|X\|_F \|y\|_2 \qquad \text{for all } y \in \mathbb{K}^{\mathcal{J}}, \tag{4.8b}$$

$$\|QX\|_F = \|X\|_F \qquad \text{for all isometric } Q \in \mathbb{K}^{\mathcal{K}\times\mathcal{I}}, \tag{4.8c}$$

$$\|XQ^*\|_F = \|X\|_F \qquad \text{for all isometric } Q \in \mathbb{K}^{\mathcal{K}\times\mathcal{J}}. \tag{4.8d}$$

Proof. Equation (4.8a) follows directly from the definition

$$\|X\|_F^2 = \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} |x_{ij}|^2 = \sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{I}} |\bar{x}_{ij}|^2 = \|X^*\|_F^2.$$

The next equation (4.8b) is a consequence of the Cauchy-Schwarz inequality, i.e.,

$$\begin{aligned}\|Xy\|_2^2 &= \sum_{i\in\mathcal{I}} |(Xy)_i|^2 = \sum_{i\in\mathcal{I}}\Big|\sum_{j\in\mathcal{J}} x_{ij}y_j\Big|^2 \leq \sum_{i\in\mathcal{I}}\Big(\sum_{j\in\mathcal{J}}|x_{ij}|^2\Big)\Big(\sum_{j\in\mathcal{J}}|y_j|^2\Big)\\ &= \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}|x_{ij}|^2\|y\|_2^2 = \|X\|_F^2\|y\|_2^2.\end{aligned}$$

For the equation (4.8c), we find

$$\begin{aligned}\|QX\|_F^2 &= \sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{J}}|(QX)_{kj}|^2 = \sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{J}}\overline{(QX)_{kj}}(QX)_{kj}\\ &= \sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{J}}\Big(\sum_{i\in\mathcal{I}}\bar{q}_{ki}\bar{x}_{ij}\Big)\Big(\sum_{\ell\in\mathcal{I}}q_{k\ell}x_{\ell j}\Big) = \sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}}\sum_{\ell\in\mathcal{I}}\bar{x}_{ij}x_{\ell j}\sum_{k\in\mathcal{K}}\bar{q}_{ki}q_{k\ell}\\ &= \sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}}\sum_{\ell\in\mathcal{I}}\bar{x}_{ij}x_{\ell j}(Q^*Q)_{i\ell} = \sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}}\bar{x}_{ij}x_{ij} = \sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}}|x_{ij}|^2 = \|X\|_F^2.\end{aligned}$$

Combining (4.8a) and (4.8c), we obtain

$$\|XQ^*\|_F = \|(XQ^*)^*\|_F = \|QX^*\|_F = \|X^*\|_F = \|X\|_F,$$

and this is the final equation (4.8d). ■

We can turn the space of matrices into a Hilbert space by introducing an inner product corresponding to the Frobenius norm.

Lemma 4.22 (Frobenius product) *The* Frobenius product *is defined by*

$$\langle X,Y\rangle_F := \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\bar{x}_{ij}y_{ij} \qquad \text{for all } X,Y\in\mathbb{K}^{\mathcal{I}\times\mathcal{J}}.$$

We have

$$\begin{aligned}\|X\|_F &= \sqrt{\langle X,X\rangle_F} && \text{for all } X\in\mathbb{K}^{\mathcal{I}\times\mathcal{J}}, && (4.9a)\\ \langle X,Y\rangle_F &= \overline{\langle X^*,Y^*\rangle_F} && \text{for all } X,Y\in\mathbb{K}^{\mathcal{I}\times\mathcal{J}}, && (4.9b)\\ \langle X,YZ\rangle_F &= \langle Y^*X,Z\rangle_F && \text{for all } X\in\mathbb{K}^{\mathcal{I}\times\mathcal{K}}, Y\in\mathbb{K}^{\mathcal{I}\times\mathcal{J}}, Z\in\mathbb{K}^{\mathcal{J}\times\mathcal{K}}, && (4.9c)\\ \langle X,YZ\rangle_F &= \langle XZ^*,Y\rangle_F && \text{for all } X\in\mathbb{K}^{\mathcal{I}\times\mathcal{K}}, Y\in\mathbb{K}^{\mathcal{I}\times\mathcal{J}}, Z\in\mathbb{K}^{\mathcal{J}\times\mathcal{K}}. && (4.9d)\end{aligned}$$

Proof. Let $X,Y\in\mathbb{K}^{\mathcal{I}\times\mathcal{J}}$. We have

$$\begin{aligned}\|X\|_F^2 &= \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}|x_{ij}|^2 = \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\bar{x}_{ij}x_{ij} = \langle X,X\rangle_F,\\ \langle X,Y\rangle_F &= \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\bar{x}_{ij}y_{ij} = \overline{\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}x_{ij}\bar{y}_{ij}} = \overline{\sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}}x_{ij}\bar{y}_{ij}} = \overline{\langle X^*,Y^*\rangle_F}.\end{aligned}$$

Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{K}}$, $Y \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$, and $Z \in \mathbb{K}^{\mathcal{J}\times\mathcal{K}}$. We have

$$\begin{aligned}\langle X, YZ\rangle_F &= \sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}} \bar{x}_{ik}(YZ)_{ik} = \sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}} \bar{x}_{ik}\sum_{j\in\mathcal{J}} y_{ij}z_{jk} = \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}} y_{ij}\bar{x}_{ik}z_{jk} \\ &= \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} \overline{(Y^*X)_{jk}}z_{jk} = \langle Y^*X, Z\rangle_F.\end{aligned}$$

Combining (4.9b) and (4.9c) yields

$$\langle X, YZ\rangle_F = \overline{\langle X^*, (YZ)^*\rangle_F} = \overline{\langle X^*, Z^*Y^*\rangle_F} = \overline{\langle ZX^*, Y^*\rangle_F} = \langle XZ^*, Y\rangle_F,$$

completing the proof. ∎

While the Frobenius norm is frequently a convenient tool for obtaining estimates for the approximation error, the bound (4.8b) can be very pessimistic. Therefore we consider the *spectral norm*, which makes this bound as sharp as possible

Definition 4.23 (Spectral norm) *The* spectral norm *of a matrix $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ is given by*

$$\|X\|_2 := \max\{\|Xy\|_2 \ : \ y \in \mathbb{K}^{\mathcal{J}} \text{ with } \|y\|_2 = 1\}.$$

Since the unit sphere $\{y \in \mathbb{K}^{\mathcal{J}} \ : \ \|y\|_2 = 1\}$ is a compact set and $y \mapsto \|Xy\|_2$ is a continuous function, the maximum exists and the norm is well-defined.

Lemma 4.24 (Spectral norm) *Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$. We have*

$$\|Xy\|_2 \le \|X\|_2\|y\|_2 \qquad \text{for all } y \in \mathbb{K}^{\mathcal{I}}, \tag{4.10a}$$

$$\|X\|_2 = \sup\left\{\frac{|z^*Xy|}{\|z\|_2\|y\|_2} \ : \ y \in \mathbb{K}^{\mathcal{J}}\setminus\{0\},\ z \in \mathbb{K}^{\mathcal{I}}\setminus\{0\}\right\}, \tag{4.10b}$$

$$\|X\|_2 = \|X^*\|_2, \tag{4.10c}$$

$$\|QX\|_2 = \|X\|_2 \qquad \text{for all isometric } Q \in \mathbb{K}^{\mathcal{K}\times\mathcal{I}}, \tag{4.10d}$$

$$\|XQ^*\|_2 = \|X\|_2 \qquad \text{for all isometric } Q \in \mathbb{K}^{\mathcal{K}\times\mathcal{J}}, \tag{4.10e}$$

$$\|D\|_2 = \max\{|d_{ii}| \ : \ i \in \mathcal{I}\} \qquad \text{for all diagonal } D \in \mathbb{K}^{\mathcal{I}\times\mathcal{I}}. \tag{4.10f}$$

Proof. Let $y \in \mathbb{K}^{\mathcal{J}}$. If $y = 0$, we have $Xy = 0$, and $\|Xy\|_2 = 0 \le \|X\|_2\|y\|_2$ holds. If $y \neq 0$, we have $\|y\|_2 \neq 0$ and can define $\hat{y} := y/\|y\|_2$, ensuring $\|\hat{y}\|_2 = 1$. Applying Definition 4.23, we obtain

$$\|Xy\|_2 = \|X\hat{y}\|_2\|y\|_2 \le \|X\|_2\|y\|_2$$

and have proven (4.10a).

To prove (4.10b), we let

$$\alpha := \sup\left\{\frac{|z^*Xy|}{\|z\|_2\|y\|_2} \ : \ y \in \mathbb{K}^{\mathcal{J}}\setminus\{0\},\ z \in \mathbb{K}^{\mathcal{I}}\setminus\{0\}\right\} \ge 0.$$

By the Cauchy-Schwarz inequality and (4.10a), we have

$$|z^*Xy| \leq \|z\|_2\|Xy\|_2 \leq \|z\|_2\|X\|_2\|y\|_2 \qquad \text{for all } y \in \mathbb{K}^{\mathcal{J}},\ z \in \mathbb{K}^{\mathcal{I}}$$

and conclude $\alpha \leq \|X\|_2$.

If $\|X\|_2 = 0$, we immediately find $\|X\|_2 \leq \alpha$. Otherwise, we choose $y \in \mathbb{K}^{\mathcal{J}}$ with $\|y\|_2 = 1$ and $\|Xy\|_2 = \|X\|_2$. Due to $\|X\|_2 > 0$, the vector $z := Xy$ is not zero, and we get

$$\alpha \geq \frac{|z^*Xy|}{\|z\|_2\|y\|_2} = \frac{|(Xy)^*(Xy)|}{\|Xy\|_2\|y\|_2} = \frac{\|Xy\|_2^2}{\|Xy\|_2\|y\|_2} = \|Xy\|_2 = \|X\|_2.$$

This proves $\alpha = \|X\|_2$, i.e., (4.10b).

Using this equation and the identity $|a^*b| = |b^*a|$ for $a, b \in \mathbb{K}^{\mathcal{J}}$, we immediately find

$$\begin{aligned}
\|X\|_2 &= \sup\left\{ \frac{|z^*Xy|}{\|z\|_2\|y\|_2} \ :\ y \in \mathbb{K}^{\mathcal{J}} \setminus \{0\},\ z \in \mathbb{K}^{\mathcal{I}} \setminus \{0\} \right\} \\
&= \sup\left\{ \frac{|(X^*z)^*y|}{\|z\|_2\|y\|_2} \ :\ y \in \mathbb{K}^{\mathcal{J}} \setminus \{0\},\ z \in \mathbb{K}^{\mathcal{I}} \setminus \{0\} \right\} \\
&= \sup\left\{ \frac{|y^*X^*z|}{\|y\|_2\|z\|_2} \ :\ z \in \mathbb{K}^{\mathcal{I}} \setminus \{0\},\ y \in \mathbb{K}^{\mathcal{J}} \setminus \{0\} \right\} = \|X^*\|_2
\end{aligned}$$

and have proven (4.10c).

Let now $Q \in \mathbb{K}^{\mathcal{K}\times\mathcal{I}}$ be isometric. Due to (4.4b), we have

$$\begin{aligned}
\|X\|_2 &= \max\{\|Xy\|_2 \ :\ y \in \mathbb{K}^{\mathcal{J}} \text{ with } \|y\|_2 = 1\} \\
&= \max\{\|QXy\|_2 \ :\ y \in \mathbb{K}^{\mathcal{J}} \text{ with } \|y\|_2 = 1\} = \|QX\|_2,
\end{aligned}$$

which proves (4.10d).

Let $Q \in \mathbb{K}^{\mathcal{K}\times\mathcal{J}}$ be isometric. Combining (4.10c) with (4.10d) yields

$$\|XQ^*\|_2 = \|(XQ^*)^*\|_2 = \|QX^*\|_2 = \|X^*\|_2 = \|X\|_2.$$

Now let $D \in \mathbb{K}^{\mathcal{I}\times\mathcal{I}}$ be a diagonal matrix, and let $\mu := \max\{|d_{ii}| \ :\ i \in \mathcal{I}\}$. We have

$$\|Dy\|_2^2 = \sum_{i\in\mathcal{I}} |d_{ii}y_i|^2 = \sum_{i\in\mathcal{I}} |d_{ii}|^2|y_i|^2 \leq \mu^2 \sum_{i\in\mathcal{I}} |y_i|^2 = \mu^2\|y\|_2^2$$

for all $y \in \mathbb{K}^{\mathcal{I}}$, and Definition 4.23 yields $\|D\|_2 \leq \mu$. We can find $j \in \mathcal{I}$ with $\mu = |d_{jj}|$. Let $\delta_j \in \mathbb{K}^{\mathcal{I}}$ denote the j-th canonical unit vector with

$$(\delta_j)_i = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i \in \mathcal{I}.$$

We have

$$\|D\delta_j\|_2^2 = \sum_{i\in\mathcal{I}} |d_{ii}(\delta_j)_i|^2 = |d_{jj}|^2 = \mu^2$$

and use Definition 4.23 to conclude $\|D\|_2 \geq \|D\delta_j\|_2 = \mu$, .e., $\|D\|_2 = \mu$. ∎

Lemma 4.25 (Approximation error) *Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$, let $p := \operatorname{rank}(X)$, let $k \in [0,\dots,p-1]$, and let $\widetilde{X}$ be constructed as in (4.7). Then we have*

$$\|X - \widetilde{X}\|_F = \Big(\sum_{\nu=k+1}^{p} \sigma_\nu^2\Big)^{1/2}, \qquad\qquad \|X - \widetilde{X}\|_2 = \sigma_{k+1}.$$

Proof. Let (U,Σ,V) be a singular value decomposition of X. We define

$$\widetilde{\Sigma} := \begin{pmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_k & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix}$$

and observe

$$\widetilde{X} = U\widetilde{\Sigma}V^*.$$

Using (4.8c), (4.8d), (4.10d) and (4.10e), we obtain

$$\begin{aligned} \|X - \widetilde{X}\|_F &= \|U\Sigma V^* - U\widetilde{\Sigma}V^*\|_F = \|U(\Sigma - \widetilde{\Sigma})V^*\|_F = \|\Sigma - \widetilde{\Sigma}\|_F, \\ \|X - \widetilde{X}\|_2 &= \|U\Sigma V^* - U\widetilde{\Sigma}V^*\|_2 = \|U(\Sigma - \widetilde{\Sigma})V^*\|_2 = \|\Sigma - \widetilde{\Sigma}\|_2. \end{aligned}$$

Due to

$$E := \Sigma - \widetilde{\Sigma} = \begin{pmatrix} 0 & & & & & \\ & \ddots & & & & \\ & & 0 & & & \\ & & & \sigma_{k+1} & & \\ & & & & \ddots & \\ & & & & & \sigma_p \end{pmatrix},$$

we immediately find

$$\|X - \widetilde{X}\|_F = \|\Sigma - \widetilde{\Sigma}\|_F = \|E\|_F = \Big(\sum_{\nu=k+1}^{p} \sigma_\nu^2\Big)^{1/2}.$$

Using (4.10f), we obtain

$$\|X - \widetilde{X}\|_2 = \|\Sigma - \widetilde{\Sigma}\|_2 = \|E\|_2 = \max\{\sigma_\nu \ :\ \nu \in [k+1:p]\} = \sigma_{k+1}.$$

■

This result allows us to control the approximation error: once the singular values have been computed, we can choose the rank k adaptively to guarantee a prescribed accuracy: to ensure the spectral norm estimate $\|X - \widetilde{X}\|_2 \le \epsilon$, we let

$$k := \begin{cases} \min\{\ell \in [0:p-1] \ :\ \sigma_{\ell+1} \le \epsilon\} & \text{if } \sigma_p \le \epsilon, \\ p & \text{otherwise,} \end{cases}$$

while the Frobenius norm estimate $\|X - \widetilde{X}\|_F \leq \epsilon$ is ensured by choosing

$$k := \min\left\{ \ell \in [0:p] \ :\ \sum_{\nu=\ell+1}^{p} \sigma_\nu^2 \leq \epsilon^2 \right\}.$$

It is frequently a good idea to ensure *relative* error bounds like $\|X - \widetilde{X}\|_2 \leq \hat{\epsilon}\|X\|_2$ or $\|X - \widetilde{X}\|_F \leq \hat{\epsilon}\|X\|_F$. Applying Lemma 4.25 to $k = 0$ yields

$$\|X\|_2 = \sigma_1, \qquad\qquad \|X\|_F = \left(\sum_{\nu=1}^{p} \sigma_\nu^2\right)^{1/2}.$$

Once the singular values have been computed, we can therefore let $\epsilon := \hat{\epsilon}\|X\|_2$ or $\epsilon := \hat{\epsilon}\|X\|_F$, respectively, and again choose the rank adaptively.

Remark 4.26 (Low-rank projection) *Let (U, Σ, V) be a singular value decomposition of a rank-p matrix $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$. Let $k \in [0:p]$ and let $\widetilde{X}$ be defined as in (4.7).*

Let $\widehat{P} \in \mathbb{K}^{p\times k}$ be given by

$$\hat{p}_{\nu\mu} := \begin{cases} 1 & \text{if } \nu = \mu, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } \nu \in [1:p],\ \mu \in [1:k].$$

This matrix is isometric and has rank k. Since U is isometric, so is $P := U\widehat{P} \in \mathbb{K}^{\mathcal{I}\times k}$, the matrix consisting of the first k columns of U.

Using $I_k \in \mathbb{K}^{k\times k}$ to denote the k-dimensional identity matrix, we have

$$\widehat{P}\widehat{P}^* = \begin{pmatrix} I_k & \\ & 0 \end{pmatrix},$$

and $\widehat{P}\widehat{P}^*$ is an orthogonal projection into the subspace $\mathbb{K}^k \times \{0\} \subseteq \mathbb{K}^p$.

The matrix $PP^* \in \mathbb{K}^{\mathcal{I}\times\mathcal{I}}$ is an orthogonal projection into the subspace range(P) *spanned by these first k columns, and we have*

$$PP^*X = U\widehat{P}\widehat{P}^*U^*U\Sigma V^* = U\widehat{P}\widehat{P}^*\Sigma V^* = U \begin{pmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_k & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix} V^* = \widetilde{X},$$

i.e., we can see $\widetilde{X}$ as the result of an orthogonal rank-k projection applied to X.

This representation is frequently useful, e.g., if we use and algorithm that constructs only U and Σ, but not V. In this case, $\widetilde{X} = PP^*X = P(X^*P)^*$ directly yields a rank-k approximation without the need for Σ or V.

The matrix $\widetilde{X}$ defined in (4.7) is not only a low-rank approximation of X, it is the *best* approximation with respect to both the spectral and the Frobenius norm. This is a useful property for theoretical investigations, since it allows us to investigate the existence of an approximation separately from its algorithmic construction.

Lemma 4.27 (Lower bound) *Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ be a matrix of rank p, and let $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p > 0$ be its singular values.*

Let $R \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ be a matrix with $k := \operatorname{rank}(R) < p$. There is a vector $z \in \mathbb{K}^{\mathcal{J}}$ with $\|z\|_2 = 1$, $Rz = 0$ and $\|Xz\|_2 \geq \sigma_{k+1}$.

Proof. (cf. [18, Theorem 2.5.3]) Let (U, Σ, V) be a singular value decomposition of X with

$$U = \begin{pmatrix} u_1 & \ldots & u_p \end{pmatrix}, \qquad \Sigma = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_p \end{pmatrix}, \qquad V = \begin{pmatrix} v_1 & \ldots & v_p \end{pmatrix}.$$

We denote the dimension of $\mathbb{K}^{\mathcal{J}}$ by $n := |\mathcal{J}|$. Let $N := \{z \in \mathbb{K}^{\mathcal{J}} \ : \ Rz = 0\}$ denote the null space of R. Due to Lemma 4.19, we have $\dim(N) \geq n - k$. Let $W := \operatorname{span}\{v_1, \ldots, v_{k+1}\}$. Since $\{v_1, \ldots, v_p\}$ is an orthonormal basis, we have $\dim(W) = k+1$.

Since both N and W are subspaces of the n-dimensional space $\mathbb{K}^{\mathcal{J}}$, their intersection cannot be trivial, i.e., we find a vector $z \in N \cap W$ with $z \neq 0$. By scaling the vector appropriately, we can ensure $\|z\|_2 = 1$.

Due to $z \in N$, we have $Rz = 0$.

Due to $z \in W$, we can find $\gamma_1, \ldots, \gamma_{k+1} \in \mathbb{K}$ such that

$$z = \gamma_1 v_1 + \ldots + \gamma_{k+1} v_{k+1}$$

holds. Since $\{v_1, \ldots, v_p\}$ is an orthonormal basis, we have

$$\begin{aligned} v_\nu^* z &= v_\nu^*(\gamma_1 v_1 + \ldots + \gamma_{k+1} v_{k+1}) = \gamma_1 v_\nu^* v_1 + \ldots + \gamma_{k+1} v_\nu^* v_{k+1} \\ &= \begin{cases} \gamma_\nu & \text{if } \nu \leq k+1, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } \nu \in [1:p] \end{aligned}$$

and

$$\begin{aligned} 1 = \|z\|_2^2 &= (\gamma_1 v_1 + \ldots + \gamma_{k+1} v_{k+1})^* z = \bar\gamma_1 v_1^* z + \ldots + \bar\gamma_{k+1} v_{k+1}^* z \\ &= \bar\gamma_1 \gamma_1 + \ldots + \bar\gamma_{k+1}\gamma_{k+1} = |\gamma_1|^2 + \ldots + |\gamma_{k+1}|^2. \end{aligned}$$

Since U is isometric, we can apply (4.10d) to obtain

$$\|Xz\|_2^2 = \|U\Sigma V^* z\|_2^2 = \|\Sigma V^* z\|_2^2 = \sum_{\nu=1}^{p} \sigma_\nu^2 |v_\nu^* z|^2 = \sum_{\nu=1}^{k+1} \sigma_\nu^2 |\gamma_\nu|^2 \geq \sigma_{k+1}^2 \sum_{\nu=1}^{k+1} |\gamma_\nu|^2 = \sigma_{k+1}^2.$$

■

A closer look reveals that this estimate already provides us with the desired result for the spectral norm.

To prove a similar result for the Frobenius norm, we will use the vector z provided by Lemma 4.27 in a rank-one update. We require an estimate for the Frobenius norm of this update.

Lemma 4.28 (Frobenius norm, projection error) *Let $Y \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ be a matrix and let $Q \in \mathbb{K}^{\mathcal{I}\times\mathcal{K}}$ be isometric. We have*

$$\|Y - QR\|_F^2 = \|Q(Q^*Y - R)\|_F^2 + \|Y - QQ^*Y\|_F^2 \qquad \text{for all } R \in \mathbb{K}^{\mathcal{K}\times\mathcal{J}}.$$

We can see that the right-hand side takes its minimum for $R = Q^*Y$.

Applying this equation to $R = 0$ and Q with only one column, we obtain

$$\begin{aligned} \|Y\|_F^2 &= \|zz^*Y\|_F^2 + \|Y - zz^*Y\|_F^2 && \text{for all } z \in \mathbb{K}^{\mathcal{I}} \text{ with } \|z\|_2 = 1,\\ \|Y\|_F^2 &= \|Yzz^*\|_F^2 + \|Y - Yzz^*\|_F^2 && \text{for all } z \in \mathbb{K}^{\mathcal{J}} \text{ with } \|z\|_2 = 1. \end{aligned}$$

Proof. Let $R \in \mathbb{K}^{\mathcal{K}\times\mathcal{J}}$. The first result is a consequence of (4.9c), since we find

$$\begin{aligned} \|Y - QR\|_F^2 &= \|Y - QQ^*Y + QQ^*Y - QR\|_F^2\\ &= \langle (Y - QQ^*Y) + Q(Q^*Y - R), (Y - QQ^*Y) + Q(Q^*Y - R)\rangle_F\\ &= \langle Y - QQ^*Y, Y - QQ^*Y\rangle_F + \langle Y - QQ^*Y, Q(Q^*Y - R)\rangle_F\\ &\quad + \langle Q(Q^*Y - R), Y - QQ^*Y\rangle_F + \langle Q(Q^*Y - R), Q(Q^*Y - R)\rangle_F\\ &= \|Y - QQ^*Y\|_F^2 + \langle Q^*(Y - QQ^*Y), Q^*Y - R\rangle_F\\ &\quad + \langle Q^*Y - R, Q^*(Y - QQ^*Y)\rangle_F + \|Q(Q^*Y - R)\|_F^2\\ &= \|Y - QQ^*Y\|_F^2 + \langle Q^*Y - Q^*QQ^*Y, Q^*Y - R\rangle_F\\ &\quad + \langle Q^*Y - R, Q^*Y - Q^*QQ^*Y\rangle_F + \|Q(Q^*Y - R)\|_F^2. \end{aligned}$$

Since Q is isometric, we have $Q^*Q = I$ and conclude

$$\begin{aligned} \|Y - QR\|_F^2 &= \|Y - QQ^*Y\|_F^2 + \langle Q^*Y - Q^*Y, Q^*Y - R\rangle_F\\ &\quad + \langle Q^*Y - R, Q^*Y - Q^*Y\rangle_F + \|Q(Q^*Y - R)\|_F^2\\ &= \|Y - QQ^*Y\|_F^2 + \|Q(Q^*Y - R)\|_F^2. \end{aligned}$$

Let now $z \in \mathbb{K}^{\mathcal{I}}$ with $\|z\|_2 = 1$. This implies $z^*z = \|z\|_2^2 = 1$, so z can be interpreted as an isometric matrix due to Notation 4.3. Applying the first equation with $R = 0$ yields

$$\|Y\|_F^2 = \|zz^*Y\|_F^2 + \|Y - zz^*Y\|_F^2.$$

For $z \in \mathbb{K}^{\mathcal{J}}$ with $\|z\|_2 = 1$, we can apply this equation in combination with (4.8a) to obtain

$$\|Y\|_F^2 = \|Y^*\|_F^2 = \|zz^*Y^*\|_F^2 + \|Y^* - zz^*Y^*\|_F^2 = \|Yzz^*\|_F^2 + \|Y - Yzz^*\|_F^2,$$

i.e., our final result. ∎

Theorem 4.29 (Best approximation) *Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ be a matrix of rank p with singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p > 0$. Let $R \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ be a matrix of rank $k \leq p$. Then we have*

$$\|X-R\|_2 \geq \begin{cases} \sigma_{k+1} & \text{if } k<p, \\ 0 & \text{otherwise}, \end{cases} \qquad \|X-R\|_F \geq \left(\sum_{\nu=k+1}^{p} \sigma_\nu^2 \right)^{1/2}.$$

Proof. (cf. [16, 28]) For the spectral norm, we use Lemma 4.27 to obtain a vector $z \in \mathbb{K}^{\mathcal{J}}$ with $\|z\|_2 = 1$, $Rz = 0$ and $\|Xz\|_2 \geq \sigma_{k+1}$. The definition of the spectral norm implies

$$\|X-R\|_2 \geq \|(X-R)z\|_2 = \|Xz - Rz\|_2 = \|Xz\|_2 \geq \sigma_{k+1}.$$

For the Frobenius norm, we prove

$$\|X-R\|_F^2 \geq \sum_{\nu=p-\ell+1}^{p} \sigma_\nu^2 \qquad \text{for all } R \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}} \text{ with } \operatorname{rank}(R) \leq p-\ell \tag{4.11}$$

by induction over $\ell \in \mathbb{N}_0$ and note that applying it to $\ell := p - \operatorname{rank}(R)$ will give us the desired result.

If $\ell = 0$ holds, the estimate's right-hand side equals zero.

Let now $\ell \in \mathbb{N}_0$ be such that (4.11) holds. Let $R \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ be a matrix with $k := \operatorname{rank}(R) \leq p-(\ell+1)$. In particular we have $k<p$ and can apply Lemma 4.27 to find $z \in \mathbb{K}^{\mathcal{J}}$ with $\|z\|_2 = 1$, $Rz = 0$ and $\|Xz\|_2 \geq \sigma_{k+1}$.

Applying Lemma 4.28 to $Y := X - R$, we get

$$\begin{aligned} \|X - R - Xzz^*\|_F^2 &= \|(X-R) - (X-R)zz^*\|_F^2 = \|Y - Yzz^*\|_F^2 = \|Y\|_F^2 - \|Yzz^*\|_F^2 \\ &= \|X-R\|_F^2 - \|(X-R)zz^*\|_F^2 = \|X-R\|_F^2 - \|Xzz^*\|_F^2. \end{aligned}$$

Due to $1 = \|z\|_2^2 = z^*z$, we can apply (4.8d) and $k \leq p-\ell-1$ to get $\|Xzz^*\|_F^2 = \|Xz\|_F^2 \geq \sigma_{k+1}^2 \geq \sigma_{p-\ell}^2$ and conclude

$$\|X-R\|_F^2 = \|X-R-Xzz^*\|_F^2 + \|Xzz^*\|_F^2 \geq \|X-R-Xzz^*\|_F^2 + \sigma_{p-\ell}^2.$$

In order to apply the induction assumption, we introduce

$$\widetilde{R} := R + Xzz^*.$$

We have $\tilde{k} := \operatorname{rank}(\widetilde{R}) \leq k+1 \leq p-(\ell+1)-1 = p-\ell$ and find

$$\|X-R\|_F^2 \geq \|X-\widetilde{R}\|_F^2 + \sigma_{p-\ell}^2 \geq \sum_{\nu=p-\ell+1}^{p} \sigma_\nu^2 + \sigma_{p-\ell}^2 = \sum_{\nu=p-\ell}^{p} \sigma_\nu^2.$$

■

Exercise 4.30 (Isometric factorization) *Let $A \in \mathbb{K}^{\mathcal{I}\times k}$ and $B \in \mathbb{K}^{\mathcal{J}\times k}$ be matrices with $k \leq |\mathcal{I}|, |\mathcal{J}|$.*

Prove that there are isometric matrices $Q_A \in \mathbb{K}^{\mathcal{I}\times k}$ and $Q_B \in \mathbb{K}^{\mathcal{J}\times k}$ and an upper triangular matrix $R \in \mathbb{K}^{k\times k}$ with $AB^* = Q_A R Q_B^*$, $Q_A^* Q_A = I$, and $Q_B^* Q_B = I$.

Hint: *A sequence of Householder reflections can be used to transform any matrix into upper triangular form.*

Exercise 4.31 (Norm) *Since we frequently use rank-k matrices to approximate other matrices, we are interested in computing the corresponding errors.*

(a) *Let $A \in \mathbb{K}^{\mathcal{I}\times k}$ and $B \in \mathbb{K}^{\mathcal{J}\times k}$ with $k \leq |\mathcal{I}|, |\mathcal{J}|$.*

Find an algorithm that takes not more than $\mathcal{O}(k^2(|\mathcal{I}|+|\mathcal{J}|))$ operations to construct a matrix $R \in \mathbb{K}^{k\times k}$ with $\|AB^*\|_2 = \|R\|_2$ and $\|AB^*\|_F = \|R\|_F$.

(b) *Let $A_1 \in \mathbb{K}^{\mathcal{I}\times k_1}$ and $B_1 \in \mathbb{K}^{\mathcal{J}\times k_1}$. Let $A_2 \in \mathbb{K}^{\mathcal{I}\times k_2}$ and $B_2 \in \mathbb{K}^{\mathcal{J}\times k_2}$.*

Assuming $k_1 + k_2 \leq |\mathcal{I}|, |\mathcal{J}|$, find an algorithm that takes not more than $\mathcal{O}(k^2(|\mathcal{I}| + |\mathcal{J}|))$ operations to construct a matrix $R \in \mathbb{K}^{(k_1+k_2)\times(k_1+k_2)}$ with $\|A_1B_1^* - A_2B_2^*\|_2 = \|R\|_2$ and $\|A_1B_1^* - A_2B_2^*\|_F = \|R\|_F$.

Exercise 4.32 (Block-diagonal matrix) *Let $\mathcal{I}_1, \mathcal{I}_2$ be disjoint sets with $\mathcal{I} = \mathcal{I}_1 \dot\cup \mathcal{I}_2$ and let $\mathcal{J}_1, \mathcal{J}_2$ be disjoint sets with $\mathcal{J} = \mathcal{J}_1 \dot\cup \mathcal{J}_2$. Let $X_1 \in \mathbb{K}^{\mathcal{I}_1\times\mathcal{J}_1}$ and $X_2 \in \mathbb{K}^{\mathcal{I}_2\times\mathcal{J}_2}$. Let*

$$X := \begin{pmatrix} X_1 & 0 \\ 0 & X_2 \end{pmatrix} \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}.$$

Prove

$$\|X\|_F = \sqrt{\|X_1\|_F^2 + \|X_2\|_F^2}, \qquad \|X\|_2 = \max\{\|X_1\|_2, \|X_2\|_2\}.$$

Exercise 4.33 (Projection) *Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ be a matrix of rank $p \in \mathbb{N}$. Following the proof of Theorem 4.6, we construct vectors $a_1, \ldots, a_p \in \mathbb{K}^{\mathcal{I}}$, $c_1, \ldots, c_p \in \mathbb{K}^{\mathcal{I}}$ and matrices $X_0, \ldots, X_p \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ as follows: let $X_0 := X$. For $k \in [1:p]$, choose $a_k, c_k \in \mathbb{K}^{\mathcal{I}}$ with $a_k \in \operatorname{range}(X_{k-1})$ and $c_k^* a_k = 1$. Let $X_k := X_{k-1} - a_k c_k^* X_{k-1}$.*

(a) *Prove* $\operatorname{range}(X_k) \subseteq \operatorname{range}(X_{k-1})$ *for all $k \in [1:p]$.*

(b) *Prove $c_k^* x = 0$ for all $x \in \operatorname{range}(X_k)$ and $k \in [1:p]$.*

(c) *Prove that $L_k := C_k^* A_k$ is a left lower triangular matrix with unit normal, where*

$$A_k := \begin{pmatrix} a_1 & \cdots & a_k \end{pmatrix}, \qquad C_k := \begin{pmatrix} c_1 & \ldots & c_k \end{pmatrix}, \qquad k \in [1:p].$$

(d) *Prove $X_k = X - \Pi_k X$ for all $k \in [1:p]$, where $\Pi_k := A_k L_k^{-1} C_k^*$ is a projection, i.e., $\Pi_k^2 = \Pi_k$.*

4.3 Rank-revealing QR factorization

The singular value decomposition allows us to obtain the optimal low-rank approximation of a given matrix $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$, but finding this decomposition is computationally expensive: standard algorithms [17] first reduce X to a bidiagonal matrix and then apply an iterative eigenvalue solver to obtain the required diagonal form. Already the first step requires computational work on the order of $n_\mathcal{I} n_\mathcal{J} \min\{n_\mathcal{I}, n_\mathcal{J}\}$ operations for a general matrix X, where we use again $n_\mathcal{I} := |\mathcal{I}|$ and $n_\mathcal{J} := |\mathcal{J}|$ to denote the cardinalities of the index sets $\mathcal{I}$ and $\mathcal{J}$.

If we no longer insist on finding the best possible low-rank approximation, we can consider alternative and less computationally expensive alternatives.

Our first approach is based on a QR factorization. In order to be able to use the standard definitions of triangular matrices, we assume $\mathcal{I} = [1 : n]$ and $\mathcal{J} = [1 : m]$ with $n, m \in \mathbb{N}$. Any matrix $X \in \mathbb{K}^{n\times m}$ can be factorized as

$$X = QR$$

with a unitary matrix $Q \in \mathbb{K}^{n\times n}$ and an upper triangular matrix $R \in \mathbb{K}^{n\times m}$. Let $p := \min\{n, m\}$. Given $k \in [1 : p]$, splitting the factors into

$$Q = \begin{pmatrix} Q_k & Q_* \end{pmatrix}, \qquad Q_k \in \mathbb{K}^{n\times k}, \qquad Q_* \in \mathbb{K}^{n\times(n-k)},$$
$$R = \begin{pmatrix} R_k \\ R_* \end{pmatrix}, \qquad R_k \in \mathbb{K}^{k\times m}, \qquad R_* \in \mathbb{K}^{(n-k)\times m}$$

yields

$$X = QR = \begin{pmatrix} Q_k & Q_* \end{pmatrix} \begin{pmatrix} R_k \\ R_* \end{pmatrix} = Q_k R_k + Q_* R_*,$$

and since Q_k has only k columns, $Q_k R_k$ can be considered a rank-k approximation of the matrix X.

Due to the identity

$$\begin{pmatrix} I_k & 0 \\ 0 & I_{n-k} \end{pmatrix} = I_n = Q^* Q = \begin{pmatrix} Q_k^* \\ Q_*^* \end{pmatrix} \begin{pmatrix} Q_k & Q_* \end{pmatrix} = \begin{pmatrix} Q_k^* Q_k & Q_k^* Q_* \\ Q_*^* Q_k & Q_*^* Q_* \end{pmatrix},$$

both the matrices Q_k and Q_* are isometric, and we can use (4.4b) to compute the error of the approximation via

$$\|X - Q_k R_k\|_2 = \|Q_* R_*\|_2 = \|R_*\|_2, \qquad \|X - Q_k R_k\|_F = \|Q_* R_*\|_F = \|R_*\|_F.$$

If we construct the QR factorization by Householder reflections, we can reduce the complexity by stopping early: if we denote the first k Householder reflections by $H_1, \ldots, H_k$, we have

$$H_k \cdots H_1 X = \begin{pmatrix} R_{kk} & R_{k*} \\ & R_{**} \end{pmatrix}$$

with $R_{kk} \in \mathbb{K}^{k\times k}$, $R_{k*} \in \mathbb{K}^{k\times(m-k)}$, and $R_{**} \in \mathbb{K}^{(n-k)\times(m-k)}$. If R_{**} is sufficiently small, we can approximate it by zero and find the rank-k factorization

$$H_k \cdots H_1 X \approx \begin{pmatrix} R_{kk} & R_{k*} \\ & 0 \end{pmatrix}, \qquad X \approx H_1^* \cdots H_k^* \begin{pmatrix} R_{kk} & R_{k*} \\ & 0 \end{pmatrix}.$$

Since the Householder reflections are orthogonal, we have

$$\left\| X - H_1^* \cdots H_k \begin{pmatrix} R_{kk} & R_{k*} \\ & 0 \end{pmatrix} \right\|_2 = \left\| H_k \cdots H_1 X - \begin{pmatrix} R_{kk} & R_{k*} \\ & 0 \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} 0 & 0 \\ & R_{**} \end{pmatrix} \right\|_2,$$

and the same holds for the Frobenius norm. If we keep track of R_{**}, we can stop applying Householder reflections as soon as the remaining error is small enough. If k steps are sufficient, the number of operations is reduced from $\mathcal{O}(nmp)$ to $\mathcal{O}(nmk)$.

Of course we would like the error to decrease as rapidly as possible. A pivoting strategy can help us achieve this goal: as long as the norm of the remainder matrix R_{**} is not small enough, we choose a column with maximal norm, swap it to the first position in the remainder, and use a Householder reflection to eliminate it. This means that this column will not contribute anything to the next remainder matrix, and we can hope that the error norms will decay rapidly.

Let us return our attention to the general case. We can enumerate the row indices $i_1, i_2, \ldots, i_n$ in any order, since the order of the indices does not matter to the Householder reflections we are going to apply. In the first step, we choose a column index $j_1 \in \mathcal{J}$ such that the norm $\|X|_{\mathcal{I}\times\{j_1\}}\|_2$ of the corresponding column is as large as possible. We apply a Householder reflection $H_1 \in \mathbb{K}^{\mathcal{I}\times\mathcal{I}}$ that eliminates all entries in this column except for the i_1-th.

The remainder matrix is now

$$(H_1 X)|_{(\mathcal{I}\setminus\{i_1\})\times(\mathcal{J}\setminus\{j_1\})},$$

and if it is not yet small enough, we repeat the procedure: we choose a column index $j_2 \in \mathcal{J} \setminus \{j_1\}$ such that $\|(H_1X)|_{(\mathcal{I}\setminus\{i_1\})\times\{j_2\}}\|_2$ is as large as possible and apply the next Householder reflection.

After k steps, we have sets $\tau_k := \{i_1, \ldots, i_k\}$ and $\sigma_k := \{j_1, \ldots, j_k\}$ of row and column indices and Householder reflections $H_1, \ldots, H_k$, and the remainder matrix is

$$(H_k \cdots H_1 X)_{(\mathcal{I}\setminus\tau_k)\times(\mathcal{J}\setminus\sigma_k)}.$$

Once the remainder is small enough, we can replace it by zero to obtain the right factor B of our low-rank representation and accumulate the Householder reflections to obtain the left factor A. This leads to the algorithm summarized in Figure 4.1.

The algorithm overwrites the matrix X successively with the results of the Householder reflections. $X|_{\tau_k\times\sigma_k}$ corresponds to the matrix R_{kk} of our derivation, $X|_{\tau_k\times(\mathcal{J}\setminus\sigma_k)}$ to the matrix R_{k*}, and $X|_{(\mathcal{I}\setminus\tau_k)\times(\mathcal{J}\setminus\sigma_k)}$ is the remainder matrix R_{**}. Once the remainder is small enough, we replace it by zero and have

$$X = \begin{pmatrix} X|_{\tau_k\times\sigma_k} & X|_{\tau_k\times(\mathcal{J}\setminus\sigma_k)} \\ & X|_{(\mathcal{I}\setminus\tau_k)\times(\mathcal{J}\setminus\sigma_k)} \end{pmatrix} \approx \begin{pmatrix} X|_{\tau_k\times\sigma_k} & X|_{\tau_k\times(\mathcal{J}\setminus\sigma_k)} \\ & 0 \end{pmatrix} = \begin{pmatrix} I_{\tau_k} \\ 0 \end{pmatrix} X|_{\tau_k\times\mathcal{J}},$$

```
procedure rrqr($X$, var $A$, $B$);
  $\tau_0 \leftarrow \emptyset$;   $\sigma_0 \leftarrow \emptyset$;   $k \leftarrow 0$;
  while $\|X|_{(\mathcal{I}\setminus\tau_k)\times(\mathcal{J}\setminus\sigma_k)}\|$ is too large do begin
    Choose an arbitrary $i_{k+1} \in \mathcal{I} \setminus \tau_k$;
    Choose $j_{k+1} \in \mathcal{J} \setminus \sigma_k$ such that $\|X|_{(\mathcal{I}\setminus\tau_k)\times\{j_{k+1}\}}\|_2$ is maximal;
    $\tau_{k+1} \leftarrow \tau_k \cup \{i_{k+1}\}$;   $\sigma_{k+1} \leftarrow \sigma_k \cup \{j_{k+1}\}$;
    Find a Householder reflection $H_{k+1} \in \mathbb{K}^{(\mathcal{I}\setminus\tau_k)\times(\mathcal{I}\setminus\tau_k)}$ such that
      $(H_{k+1}X|_{(\mathcal{I}\setminus\tau_k)\times\{j_{k+1}\}})|_{\mathcal{I}\setminus(\tau_{k+1})} = 0$;
    $X|_{(\mathcal{I}\setminus\tau_k)\times(\mathcal{J}\setminus\sigma_k)} \leftarrow H_{k+1}X|_{(\mathcal{I}\setminus\tau_k)\times(\mathcal{J}\setminus\sigma_k)}$;
    $k \leftarrow k+1$
  end;
  $B \leftarrow X|^*_{\tau_k\times\mathcal{J}}$;   $A \leftarrow \begin{pmatrix} I_{\tau_k} \\ 0 \end{pmatrix} \in \mathbb{K}^{\mathcal{I}\times\tau_k}$;
  for $\nu = k$ downto 1 do
    $A|_{(\mathcal{I}\setminus\tau_\nu)\times\tau_k} \leftarrow H_\nu^* A|_{(\mathcal{I}\setminus\tau_\nu)\times\tau_k}$
end
```

Figure 4.1: Rank-revealing QR factorization

where $I_{\tau_k} \in \mathbb{K}^{\tau_k\times\tau_k}$ denotes the identity matrix, and we only have to apply the Householder reflections to the left factor to obtain the rank-k approximation $AB^* = Q_kR_k$.

4.4 Rank-revealing LR factorization and cross approximation

The rank-revealing QR factorization can be significantly faster than the singular value decomposition, but since applying even the first reflection requires at least $|\mathcal{I}|\,|\mathcal{J}|$ operations, we cannot hope to get less than quadratic complexity.

Sacrificing some of the stability provided by the unitary transformations, we can further reduce the computational work: instead of a QR factorization, we employ an LR factorization.

We start by considering a matrix $X \in \mathbb{K}^{n\times n}$ that has an LR factorization

$$X = LR$$

with a lower triangular matrix $L \in \mathbb{K}^{n\times n}$ and an upper triangular matrix $R \in \mathbb{K}^{n\times n}$. Given $k \in [1:n]$, we can split the factors into

$$L = \begin{pmatrix} L_{kk} & \\ L_{*k} & L_{**} \end{pmatrix}, \qquad L_{kk} \in \mathbb{K}^{k\times k}, \qquad L_{*k} \in \mathbb{K}^{(n-k)\times k}, \qquad L_{**} \in \mathbb{K}^{(n-k)\times(n-k)},$$
$$R = \begin{pmatrix} R_{kk} & R_{k*} \\ & R_{**} \end{pmatrix}, \qquad R_{kk} \in \mathbb{K}^{k\times k}, \qquad R_{k*} \in \mathbb{K}^{k\times(n-k)}, \qquad R_{**} \in \mathbb{K}^{(n-k)\times(n-k)}.$$

As in the case of the QR factorization, we use the first k columns of L and the first k rows of R for our approximation, i.e.,

$$X \approx \widetilde{X} := \begin{pmatrix} L_{kk} \\ L_{*k} \end{pmatrix} \begin{pmatrix} R_{kk} & R_{k*} \end{pmatrix} = \begin{pmatrix} L_{kk}R_{kk} & L_{kk}R_{k*} \\ L_{*k}R_{kk} & L_{*k}R_{k*} \end{pmatrix}.$$

Splitting X accordingly, i.e., into

$$X = \begin{pmatrix} X_{kk} & X_{k*} \\ X_{*k} & X_{**} \end{pmatrix}, \qquad X_{kk} \in \mathbb{K}^{k\times k}, \qquad X_{k*} \in \mathbb{K}^{k\times(n-k)},$$
$$X_{*k} \in \mathbb{K}^{(n-k)\times k}, \qquad X_{**} \in \mathbb{K}^{(n-k)\times(n-k)},$$

we find

$$\begin{aligned} \begin{pmatrix} X_{kk} & X_{k*} \\ X_{*k} & X_{**} \end{pmatrix} &= X = LR = \begin{pmatrix} L_{kk} & \\ L_{*k} & L_{**} \end{pmatrix} \begin{pmatrix} R_{kk} & R_{k*} \\ & R_{**} \end{pmatrix} \\ &= \begin{pmatrix} L_{kk}R_{kk} & L_{kk}R_{k*} \\ L_{*k}R_{kk} & L_{*k}R_{k*} + L_{**}R_{**} \end{pmatrix}, \end{aligned} \tag{4.12}$$

i.e., the approximation error is given by

$$X - \widetilde{X} = \begin{pmatrix} 0 & 0 \\ 0 & L_{**}R_{**} \end{pmatrix}. \tag{4.13}$$

We also obtain the equations

$$L_{kk}R_{kk} = X_{kk}, \qquad L_{kk}R_{k*} = X_{k*}, \qquad L_{*k}R_{kk} = X_{*k},$$

i.e., we can compute the approximation $\widetilde{X}$ using only the first k rows and columns of X.

Finding the LR factorization of X_{kk} takes not more than k^3 operations, solving $L_{kk}R_{k*} = X_{k*}$ by forward substitution takes not more than $k^2(n-k)$ operations, while solving $L_{*k}R_{kk} = X_{*k}$ by forward substitution (it is equivalent to $R_{kk}^* L_{*k}^* = X_{*k}^*$ with the lower triangular matrix R_{kk}^*) also takes not more than $k^2(n-k)$ operations, so a total of less than $2k^2n$ operations are sufficient to obtain $\widetilde{X}$.

Remark 4.34 (Comparison with rank-revealing QR) *To compare the rank-revealing LR factorization with the rank-revealing QR factorization, we assume that the first k columns of X are sufficient to approximate the entire matrix.*

In this case, we can construct the first k Householder reflections in $\mathcal{O}(nk^2)$ operations, since we only have to apply them to the first k columns. This gives us the factor Q of the QR factorization. So far, rank-revealing QR and LR are comparable.

Unfortunately, computing the remaining $n-k$ columns of the factor $R = Q^*X$ requires $\mathcal{O}(n(n-k)k)$ operations, i.e., the rank-revealing QR factorization has quadratic complexity with respect to n, even under these particularly convenient conditions.

As in the case of the QR factorization, we can construct the LR factorization inductively by increasing the rank: applying (4.12) to $k = 1$, we find

$$L_{11}R_{11} = X_{11}, \qquad L_{11}R_{1*} = X_{1*}, \qquad L_{*1}R_{11} = X_{*1}, \qquad L_{*1}R_{1*} + L_{**}R_{**} = X_{**}.$$

As long as $X_{11} \neq 0$ holds, we can satisfy the first equation by choosing $L_{11} = 1$, $R_{11} = X_{11}$ and obtaining $R_{1*} = X_{1*}$, $L_{*1} = X_{*1}/X_{11}$, and

$$L_{**}R_{**} = X_{**} - L_{*1}R_{1*}.$$

We have found the first row and column of L and R and can proceed by looking for the LR factorization of the *Schur complement*

$$X_{**} - L_{*1}R_{1*} = X_{**} - X_{*1}X_{11}^{-1}X_{1*}$$

to construct as many rows and columns as we need.

The construction breaks down if we encounter $X_{11} = 0$, but as long as $X \neq 0$ holds, we can fix this issue by applying row and column permutations that move one of the non-zero coefficients into the first row and column.

Now we have to consider the generalization of our approach to general matrices $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$. This leads to a class of algorithms known as *cross approximation* methods [32, 19, 33, 1, 4].

We construct sets of row pivot indices $\tau_k := \{i_1, \dots, i_k\} \subseteq \mathcal{I}$ and column pivot indices $\sigma_k := \{j_1, \dots, j_k\} \subseteq \mathcal{J}$ and apply the standard LR factorization to the matrix $\widehat{X} \in \mathbb{K}^{k\times k}$ with $\hat{x}_{\nu\mu} = x_{i_\nu, j_\mu}$. The pivot indices are constructed during the LR factorization algorithm in order to avoid encountering a zero on the diagonal.

We denote the ν-th column of L by $\ell^{(\nu)} \in \mathbb{K}^{\mathcal{I}}$ and the ν-th row of R by $r^{(\nu)} \in \mathbb{K}^{\mathcal{J}}$.

We start by choosing indices $i_1 \in \mathcal{I}$ and $j_1 \in \mathcal{J}$ such that $x_{i_1,j_1} \neq 0$ holds. The first column of L and the first row of R are given by

$$r_j^{(1)} = \bar{x}_{i_1,j}, \qquad \ell_i^{(1)} = \frac{x_{i,j_1}}{x_{i_1,j_1}} \qquad \text{for all } i \in \mathcal{I},\ j \in \mathcal{J}.$$

To proceed, we construct the remainder, i.e., the first Schur complement

$$X^{(1)} = X - \ell^{(1)}(r^{(1)})^*$$

and look to approximate its LR factorization. The resulting algorithm is shown in Figure 4.2.

It constructs matrices

$$L = \begin{pmatrix} \ell^{(1)} & \cdots & \ell^{(k)} \end{pmatrix} \in \mathbb{K}^{\mathcal{I}\times[1:k]}, \qquad R := \begin{pmatrix} (r^{(1)})^* \\ \vdots \\ (r^{(k)})^* \end{pmatrix} \in \mathbb{K}^{[1:k]\times\mathcal{J}}$$

such that

$$X = LR + X^{(k)}$$

```
procedure aca($X$, var $L$, $R$);
  $\tau_0 \leftarrow \emptyset$;   $\sigma_0 \leftarrow \emptyset$;   $X^{(0)} \leftarrow X$;   $k \leftarrow 0$;
  while $\|X^{(k)}\|$ is too large do begin
    Choose $i_{k+1} \in \mathcal{I} \setminus \tau_k$ and $j_{k+1} \in \mathcal{J} \setminus \sigma_k$ with $x^{(k)}_{i_{k+1},j_{k+1}} \neq 0$;
    for $j \in \mathcal{J}$ do $r^{(k+1)}_j \leftarrow \bar{x}^{(k)}_{i_{k+1},j}$;
    for $i \in \mathcal{I}$ do $\ell^{(k+1)}_i \leftarrow x^{(k)}_{i,j_{k+1}} / \bar{r}^{(k+1)}_{j_{k+1}}$;
    $X^{(k+1)} \leftarrow X^{(k)} - \ell^{(k+1)}(r^{(k+1)})^*$;
    $\tau_{k+1} \leftarrow \tau_k \cup \{i_{k+1}\}$;   $\sigma_{k+1} \leftarrow \sigma_k \cup \{j_{k+1}\}$;
    $k \leftarrow k+1$
  end
end
```

Figure 4.2: Adaptive cross approximation

holds, and the algorithm stops as soon as the error $X^{(k)} = X - LR$ is sufficiently small, i.e., we can use

$$\widetilde{X} := LR = \sum_{\nu=1}^{k} \ell^{(\nu)}(r^{(\nu)})^*$$

as a rank-k approximation of X.

Remark 4.35 (Rank reduction) *This construction follows the pattern of Lemma 4.4: $a := \ell^{(1)}$ is a vector in $\operatorname{range}(X)$. Let $b \in \mathbb{K}^{\mathcal{I}}$ be the canonical unit vector equal to one in the i_1-th component and equal to zero everywhere else. We have $b^*a = \ell^{(1)}_{i_1} = 1$, and we can apply Lemma 4.4 to find that the rank of*

$$X - ab^*X = X - \ell^{(1)}(r^{(1)})^* = X^{(1)}$$

is one smaller than the rank of X.

Lemma 4.36 (Cross approximation) *We have*

$$X^{(\nu)}|_{\tau_\nu \times \mathcal{J}} = 0, \qquad X^{(\nu)}|_{\mathcal{I} \times \sigma_\nu} = 0 \qquad \text{for all } \nu \in [0:k]$$

and

$$\begin{aligned} \ell^{(\nu)}_{i_\nu} &= 1, & \ell^{(\nu)}_{i_\mu} &= 0 & &\text{for all } \nu \in [1:k],\ \mu \in [1:\nu-1] \\ r^{(\nu)}_{j_\nu} &\neq 0, & r^{(\nu)}_{j_\mu} &= 0 & &\text{for all } \nu \in [1:k],\ \mu \in [1:\nu-1]. \end{aligned}$$

Proof. We prove the first claim by induction.

For $\nu = 0$, we have $\tau_0 = \emptyset$ and $\sigma_\nu = \emptyset$, so there is nothing to prove.

Let now $\nu \in [0:k-1]$ be given with $X^{(\nu)}|_{\tau_\nu \times \mathcal{J}} = 0$ and $X^{(\nu)}|_{\mathcal{I} \times \sigma_\nu} = 0$.

Since $\ell^{(\nu+1)}$ and $r^{(\nu+1)}$ are just a scaled column and row of $X^{(\nu)}$, our assumption immediately yields

$$\ell^{(\nu+1)}|_{\tau_\nu} = 0, \qquad r^{(\nu+1)}|_{\sigma_\nu} = 0 \tag{4.14}$$

and thus

$$\begin{aligned} X^{(\nu+1)}|_{\tau_\nu \times \mathcal{J}} &= X^{(\nu)}|_{\tau_\nu \times \mathcal{J}} - \ell^{(\nu+1)}|_{\tau_\nu} (r^{(\nu+1)})^* = 0, \\ X^{(\nu+1)}|_{\mathcal{I} \times \sigma_\nu} &= X^{(\nu)}|_{\mathcal{I} \times \sigma_\nu} - \ell^{(\nu+1)} (r^{(\nu+1)}|_{\sigma_\nu})^* = 0. \end{aligned}$$

Due to $\tau_{\nu+1} = \tau_\nu \cup \{i_{\nu+1}\}$ and $\sigma_{\nu+1} = \sigma_\nu \cup \{j_{\nu+1}\}$, we only have to observe

$$\begin{aligned} x^{(\nu+1)}_{i,j_{\nu+1}} &= x^{(\nu)}_{i,j_{\nu+1}} - \ell^{(\nu+1)}_i r^{(\nu+1)}_{j_{\nu+1}} = x^{(\nu)}_{i,j_{\nu+1}} - \frac{x^{(\nu)}_{i,j_{\nu+1}}}{x^{(\nu)}_{i_{\nu+1},j_{\nu+1}}} x^{(\nu)}_{i_{\nu+1},j_{\nu+1}} = 0 && \text{for all } i \in \mathcal{I}, \\ x^{(\nu+1)}_{i_{\nu+1},j} &= x^{(\nu)}_{i_{\nu+1},j} - \ell^{(\nu+1)}_{i_{\nu+1}} r^{(\nu+1)}_j = x^{(\nu)}_{i_{\nu+1},j} - \frac{x^{(\nu)}_{i_{\nu+1},j_{\nu+1}}}{x^{(\nu)}_{i_{\nu+1},j_{\nu+1}}} x^{(\nu)}_{i_{\nu+1},j} = 0 && \text{for all } j \in \mathcal{J}, \end{aligned}$$

to complete the induction.

The second claim follows from (4.14) and

$$\ell^{(\nu)}_{i_\nu} = \frac{x^{(\nu)}_{i_\nu,j_\nu}}{x^{(\nu)}_{i_\nu,j_\nu}} = 1, \qquad r^{(\nu)}_{j_\nu} = x^{(\nu)}_{i_\nu,j_\nu} \neq 0 \qquad \text{for all } \nu \in [1:k].$$

■

We define matrices $\widehat{L}, \widehat{R} \in \mathbb{K}^{k \times k}$ via

$$\hat{\ell}_{\nu\mu} = \ell^{(\mu)}_{i_\nu} = \ell_{i_\nu,\mu}, \qquad \hat{r}_{\nu\mu} = r^{(\nu)}_{j_\mu} = r_{\nu,j_\mu} \qquad \text{for all } \nu, \mu \in [1:k].$$

Lemma 4.36 implies that $\widehat{L}$ is a lower triangular matrix with unit diagonal, while $\widehat{R}$ is an upper triangular matrix with non-zero diagonal, i.e., both are invertible, and therefore so are $L|_{\tau_k \times [1:k]}$ and $R|_{[1:k] \times \sigma_k}$.

Lemma 4.37 (Factorized representation) *The matrix $X|_{\tau_k \times \sigma_k}$ is invertible and the approximation $\widetilde{X}$ can be represented as*

$$\widetilde{X} = LR = X|_{\mathcal{I} \times \sigma_k} X|^{-1}_{\tau_k \times \sigma_k} X|_{\tau_k \times \mathcal{J}}.$$

Proof. (see also [3, Lemma 5.1]) Due to Lemma 4.36, we have

$$X|_{\tau_k \times \mathcal{J}} = X^{(k)}|_{\tau_k \times \mathcal{J}} + L|_{\tau_k \times [1:k]} R|_{[1:k] \times \mathcal{J}} = L|_{\tau_k \times [1:k]} R, \tag{4.15a}$$

$$X|_{\mathcal{I} \times \sigma_k} = X^{(k)}|_{\mathcal{I} \times \sigma_k} + L|_{\mathcal{I} \times [1:k]} R|_{[1:k] \times \sigma_k} = LR|_{[1:k] \times \sigma_k}. \tag{4.15b}$$

Restricting the first equation (4.15a) to columns in σ_k yields

$$X|_{\tau_k \times \sigma_k} = L|_{\tau_k \times [1:k]} R|_{[1:k] \times \sigma_k}.$$

Since $L|_{\tau_k\times[1:k]}$ and $R|_{[1:k]\times\sigma_k}$ are invertible, the same holds for their product $X|_{\tau_k\times\sigma_k}$.

Multiplying (4.15a) by $L|_{\tau_k\times[1:k]}^{-1}$ from the left and (4.15b) by $R|_{[1:k]\times\sigma_k}^{-1}$ from the right gives us

$$\begin{aligned} LR &= X|_{\mathcal{I}\times\sigma_k} R|_{[1:k]\times\sigma_k}^{-1} L|_{\tau_k\times[1:k]}^{-1} X|_{\tau_k\times\mathcal{J}} \\ &= X|_{\mathcal{I}\times\sigma_k} (L|_{\tau_k\times[1:k]} R|_{[1:k]\times\sigma_k})^{-1} X|_{\tau_k\times\mathcal{J}} = X|_{\mathcal{I}\times\sigma_k} X|_{\tau_k\times\sigma_k}^{-1} X|_{\tau_k\times\mathcal{J}}. \end{aligned}$$

This is the equation we need. ■

Remark 4.38 (Partial evaluation) *In a practical implementation, we would like to compute only the coefficients of X that are required to obtain L and R. This means that we cannot compute the entire matrices $X^{(\nu)}$ for all $\nu \in [0:k]$.*

Our algorithm ensures

$$X^{(\nu)} = X - L|_{\mathcal{I}\times[1:\nu]} R|_{[1:\nu]\times\mathcal{J}},$$

so we have

$$\begin{aligned} X^{(\nu)}|_{\mathcal{I}\times\{j\}} &= X|_{\mathcal{I}\times\{j\}} - L|_{\mathcal{I}\times[1:\nu]} R|_{[1:\nu]\times\{j\}} \\ &= X|_{\mathcal{I}\times\{j\}} - \sum_{\mu=1}^{\nu} \ell^{(\mu)}\, \bar{r}_j^{(\mu)} && \text{for all } \nu \in [0:k],\ j \in \mathcal{J}, \\ X^{(\nu)}|_{\{i\}\times\mathcal{J}} &= X|_{\{i\}\times\mathcal{J}} - L|_{\{i\}\times[1:\nu]} R|_{[1:\nu]\times\mathcal{J}} \\ &= X|_{\{i\}\times\mathcal{J}} - \sum_{\mu=1}^{\nu} \ell_i^{(\mu)}\, (r^{(\mu)})^* && \text{for all } \nu \in [0:k],\ i \in \mathcal{I}, \end{aligned}$$

and this allows us to construct the required rows and columns of $X^{(\nu)}$ based only on the corresponding rows and columns of the original matrix X and the previously computed rows and columns of L and R. A typical implementation of the adaptive cross approximation algorithm requires only a way to obtain single rows and columns of the matrix that has to be approximated.

In a practical implementation, we have to address two aspects of the cross approximation that have not been discussed so far: we have to specify a stopping criterion, and we have to describe how the pivot elements $i_1, \ldots, i_k$ and $j_1, \ldots, j_k$ are chosen.

Remark 4.39 (Stopping criterion) *If we construct the full matrices $X^{(k)}$, we can simply use the Frobenius or spectral norm to determine when to stop the algorithm. Finding a* reliable *stopping criterion becomes significantly more challenging if we follow the partial evalution approach described in Remark 4.38: since we cannot check the entire matrix, we run the risk of missing large entries in the remainder matrix that are outside of the rows and columns we have chosen so far.*

Remark 4.40 (Pivot strategy) *The strategy employed to choose the pivot indices $i_1, \dots, i_k$ and $j_1, \dots, j_k$ is of particular importance if the partial evaluation approach of Remark 4.38 is employed: the algorithm only "sees" a very small part of the matrix and still has to choose suitable pivot elements that yield fast convergence and ensure that the stopping criterion is not triggered prematurely.*

Simple pivoting strategies may lead to completely inaccurate approximations [7, Example 2.2]. It is possible to prove the cross approximation algorithm will produce good results if the pivot indices are chosen correctly [19], but the corresponding strategy is computationally expensive. There are attempts [3] to derive efficient and reliable pivoting strategies, but so far they rely on additional stability assumptions.

Example 4.41 (Rank-one matrix) *Let $n \in \mathbb{N}$. We consider the funktion*

$$g\colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}, \qquad (x, y) \mapsto \begin{cases} \frac{\sin(x_1)\sin(y_1)}{\|x-y\|_2} & \text{if } x \neq y, \\ 0 & \text{otherwise.} \end{cases}$$

We can see that the function is analytic as long as we stay away from the diagonal $x = y$, and the techniques of Section 3.8 can be used to prove that interpolation converges.

Let $i_0, j_0 \in [1 : n]$. We consider the points

$$x_i := \begin{cases} (\pi/2, i/n) & \text{if } i = i_0, \\ (0, i/n) & \text{otherwise} \end{cases} \qquad \text{for all } i \in [1 : n],$$

$$y_j := \begin{cases} (3\pi/2, j/n) & \text{if } j = j_0, \\ (2\pi, j/n) & \text{otherwise} \end{cases} \qquad \text{for all } j \in [1 : n].$$

The matrix $G \in \mathbb{R}^{n \times n}$ defined by

$$g_{ij} = g(x_i, y_j) \qquad \text{for all } i, j \in [1 : n]$$

satisfies

$$g_{ij} = \begin{cases} -1/\|x_i - y_j\|_2 & \text{if } i = i_0 \text{ and } j = j_0, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i, j \in [1 : n],$$

i.e., only one coefficient in the entire matrix differs from zero, and we can choose its position arbitrarily. It is hard to imagine a pivot strategy that is able to reliably find this one coefficient without checking the entire matrix.

Remark 4.42 (Error analysis) *The error analysis of the adaptive cross approximation algorithm is discussed in [4] for boundary element matrices resulting from collocation methods and in [3] for Galerkin methods.*

The results in [1], however, should be approached with care, since they appear to be based on circular reasoning[1]. *The book [2] presents improved pivot strategies.*

[1]The polynomials p_k appearing in the proof of [1, Lemma 4] can only be defined if the Lagrange interpolation problem can be solved, and this is what has to be proven.

Remark 4.43 (Interpolation) *The result of Lemma 4.37 can be interpreted as the result of "algebraic interpolation": we define the matrix*

$$V := X|_{\mathcal{I}\times\sigma_k} X|_{\tau_k\times\sigma_k}^{-1}$$

and denote its columns by $v^{(\nu)}$ for $\nu \in \tau_k$. Due to

$$V|_{\tau_k\times\tau_k} = X|_{\tau_k\times\sigma_k} X|_{\tau_k\times\sigma_k}^{-1} = I,$$

we have

$$v^{(\nu)}_\mu = v_{\mu\nu} = \begin{cases} 1 & \text{if } \nu = \mu, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } \nu, \mu \in \tau,$$

so we can interprete the vectors v_ν as a "Lagrange basis" of the range of V corresponding to the "interpolation points" ν. The algebraic interpolation operator for this basis and these points is given by

$$\mathfrak{I} : \mathbb{K}^{\mathcal{I}} \to \mathbb{K}^{\mathcal{I}}, \qquad x \mapsto \sum_{\nu\in\tau_k} v^{(\nu)} x_\nu = V x|_{\tau_k}.$$

We have

$$\mathfrak{I}X = VX|_{\tau_k\times\mathcal{J}} = X|_{\mathcal{I}\times\sigma_k} X|_{\tau_k\times\sigma_k}^{-1} X|_{\tau_k\times\mathcal{I}} = \widetilde{X},$$

i.e., the cross approximation matrix $\widetilde{X}$ is the result of algebraic interpolation.

For any vector $x \in \mathbb{K}^{\mathcal{I}}$ we have

$$\mathfrak{I}^2 x = V(Vx|_{\tau_k})|_{\tau_k} = VX|_{\tau_k\times\sigma_k} X|_{\tau_k\times\sigma_k}^{-1} x|_{\tau_k} = Vx|_{\tau_k} = \mathfrak{I}x,$$

therefore $\mathfrak{I}$ is a projection into the range of V. In general, it is not *an orthogonal projection, and its stability, i.e., whether $\|\mathfrak{I}\|$ is bounded, depends crucially on the pivot strategy.*

If all *elements in the remainder matrix are checked and the one with the maximal value is chosen, a simple estimate for $\|\mathfrak{I}\|$ can be found in [3, Lemma 5.3].*

4.5 Hybrid cross approximation

In order to avoid the dubious reliability of partial cross approximation techniques, we can combine it with a reliable compression scheme that leads to a matrix that is small enough to apply adaptive cross approximation *without partial evaluation*, so that we can use a full pivot search and obtain guaranteed error bounds.

The *hybrid cross approximation* method (HCA, cf. [7]) relies on interpolation: we once more consider a matrix $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ given by

$$g_{ij} = g(x_i, y_j) \qquad \text{for all } i \in \mathcal{I},\ j \in \mathcal{J}$$

with a suitable kernel function g and points $(x_i)_{i\in\mathcal{I}}$ and $(y_j)_{j\in\mathcal{J}}$ in $\mathbb{R}^d$.

Given an admissible pair (t,s) of clusters $t \in \mathcal{T}_{\mathcal{I}}$, $s \in \mathcal{T}_{\mathcal{J}}$, we choose interpolation points $(\xi_{t,\nu})_{\nu\in M}$ in t and and $(\xi_{s,\mu})_{\mu\in M}$ in s with corresponding Lagrange polynomials $(\ell_{t,\nu})_{\nu\in M}$ and $(\ell_{s,\mu})_{\mu\in M}$. In a first step, we approximate g by interpolation to obtain

$$\tilde{g}(x,y) = \sum_{\nu\in M}\sum_{\mu\in M} \ell_{t,\nu}(x)\, g(\xi_{t,\nu},\xi_{s,\mu})\, \ell_{s,\mu}(y) \qquad \text{for all } x\in t,\ y\in s. \tag{4.16}$$

The matrix $S \in \mathbb{K}^{M\times M}$ given by

$$s_{\nu\mu} := g(\xi_{t,\nu},\xi_{s,\mu}) \qquad \text{for all } \nu,\mu\in M$$

is sufficiently small to allow us to approximate it by adaptive cross approximation, i.e., to find $k\in\mathbb{N}$ and $A\in\mathbb{K}^{M\times[1:k]}$, $B\in\mathbb{K}^{[1:k]\times M}$ such that

$$S \approx AB.$$

Replacing $s_{\nu\mu}$ by this approximation in (4.16) yields

$$\begin{aligned}
g(x,y) \approx \tilde{g}(x,y) &= \sum_{\nu\in M}\sum_{\mu\in M} \ell_{t,\nu}(x)\, s_{\nu\mu}\, \ell_{s,\mu}(y)\\
&\approx \sum_{\nu\in M}\sum_{\mu\in M} \ell_{t,\nu}(x) \left(\sum_{\lambda=1}^{k} a_{\nu\lambda} b_{\lambda\mu}\right) \ell_{s,\mu}(y)\\
&= \sum_{\lambda=1}^{k} \left(\sum_{\nu\in M} a_{\nu\lambda}\, \ell_{t,\nu}(x)\right)\left(\sum_{\mu\in M} b_{\lambda\mu}\, \ell_{s,\mu}(y)\right) \qquad \text{for all } x\in t,\ y\in s.
\end{aligned}$$

This is a degenerate approximation of g that consists only of k terms instead of $|M|$, and in certain applications k can be significantly smaller than $|M|$.

Still, having to evaluate the Lagrange polynomials $\ell_{t,\nu}$ and $\ell_{s,\mu}$ for all $\nu,\mu\in M$ makes working with this approximation a little cumbersome. We can use Lemma 4.37 to add a third approximation step that significantly reduces the complexity: we have

$$S \approx AB = S|_{M\times\sigma_k} S|_{\tau_k\times\sigma_k}^{-1} S|_{\tau_k\times M}$$

for the chosen sets $\tau_k, \sigma_k \subseteq M$ of row and column pivots.

Introducing $C := S|_{\tau_k\times\sigma_k}^{-1}$, we find

$$\begin{aligned}
g(x,y) \approx \tilde{g}(x,y) &= \sum_{\nu\in M}\sum_{\mu\in M} \ell_{t,\nu}(x) s_{\nu\mu} \ell_{s,\mu}(y)\\
&\approx \sum_{\nu\in M}\sum_{\mu\in M} \ell_{t,\nu}(x) (S|_{M\times\sigma_k}\, C\, S|_{\tau_k\times M})_{\nu\mu} \ell_{s,\mu}(y)\\
&= \sum_{\nu\in M}\sum_{\mu\in M} \ell_{t,\nu}(x) \left(\sum_{\lambda\in\sigma_k}\sum_{\kappa\in\tau_k} s_{\nu\lambda}\, c_{\lambda\kappa}\, s_{\kappa\mu}\right) \ell_{s,\mu}(y)
\end{aligned}$$

$$= \sum_{\lambda\in\sigma_k}\sum_{\kappa\in\tau_k}\left(\sum_{\nu\in M} s_{\nu\lambda}\,\ell_{t,\nu}(x)\right)c_{\lambda\kappa}\left(\sum_{\mu\in M} s_{\kappa\mu}\,\ell_{s,\mu}(y)\right)$$
$$= \sum_{\lambda\in\sigma_k}\sum_{\kappa\in\tau_k}\left(\sum_{\nu\in M} g(\xi_{t,\nu},\xi_{s,\lambda})\ell_{t,\nu}(x)\right)c_{\lambda\kappa}\left(\sum_{\mu\in M} g(\xi_{t,\kappa},\xi_{s,\mu})\ell_{s,\mu}(y)\right).$$

We can see that the sums over ν and μ represent interpolating polynomials corresponding to $g(\cdot,\xi_{s,\lambda})$ and $g(\xi_{t,\kappa},\cdot)$, respectively, and we can "reverse" the interpolation by substituting the kernel functions, i.e.,

$$\sum_{\nu\in M} g(\xi_{t,\nu},\xi_{s,\lambda})\ell_{t,\nu}(x) \approx g(x,\xi_{s,\lambda}) \qquad \text{for all } x\in t,\ \lambda\in\sigma_k,$$
$$\sum_{\mu\in M} g(\xi_{t,\kappa},\xi_{s,\mu})\ell_{s,\mu}(y) \approx g(\xi_{t,\kappa},y) \qquad \text{for all } y\in s,\ \kappa\in\tau_k.$$

This approach gives rise to the final degenerate approximation

$$g(x,y)\approx \tilde g_{\mathrm{hca}}(x,y) := \sum_{\lambda\in\sigma_k}\sum_{\kappa\in\tau_k} g(x,\xi_{s,\lambda})\,c_{\lambda\kappa}\,g(\xi_{t,\kappa},y) \qquad \text{for all } x\in B_t,\ y\in B_s$$

that can be evaluated without the need for Lagrange polynomials. Since the cross approximation algorithm provides us with an LR factorization of $S|_{\tau_k\times\sigma_k}$, the multiplication with $C = S|_{\tau_k\times\sigma_k}^{-1}$ can be handled by standard forward and backward substitution.

Remark 4.44 (Cross approximation) *We have seen in Lemma 4.36 that the standard cross approximation algorithm matches the pivot row and columns of the original matrix exactly. The hybrid cross approximation does the same for the pivot interpolation points and the original kernel function: for all $\nu\in\tau_k$ and $y\in s$, we have*

$$\tilde g_{hca}(\xi_{t,\nu},y) = \sum_{\lambda\in\sigma_k}\sum_{\kappa\in\tau_k} g(\xi_{t,\nu},\xi_{s,\lambda})\,c_{\lambda k}\,g(\xi_{t,\kappa},y) = \sum_{\lambda\in\sigma_k}\sum_{\kappa\in\tau_k} s_{\nu\lambda}\,c_{\lambda\kappa}\,g(\xi_{t,\kappa},y)$$
$$= \sum_{\kappa\in\tau_k}(S|_{\tau_k\times\sigma_k}C)_{\nu\kappa}\,g(\xi_{t,\kappa},y) = \sum_{\kappa\in\tau_k} I_{\nu\kappa}\,g(\xi_{t,\kappa},y) = g(\xi_{t,\nu},y),$$

and we can also prove $\tilde g_{hca}(x,\xi_{s,\mu}) = g(x,\xi_{s,\mu})$ for all $\mu\in\sigma_k$ and $x\in t$.

4.6 Global norm estimates

We have seen that we can control the error resulting from the low-rank approximation of a matrix. In a *hierarchical* matrix, we use low-rank approximations in all admissible blocks, and we have to ensure that error estimates for these blocks lead to reliable error estimates for the entire matrix.

Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a block tree, let $\mathcal{L}_{\mathcal{I}\times\mathcal{J}}$ denote the set of its leaves.

If we can bound a norm of a matrix $X\in\mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ in terms of the norms of $X|_{\hat t\times\hat s}$, $b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}$, we can apply this bound to the error matrix to control the global error. For the Frobenius norm, this bound is particularly easy to find.

Lemma 4.45 (Global Frobenius norm) *Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$. We have*

$$\|X\|_F = \left(\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \|X|_{\hat{t}\times\hat{s}}\|_F^2 \right)^{1/2}.$$

Proof. Due to Corollary 3.23, we have

$$\|X\|_F^2 = \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} |x_{ij}|^2 = \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \sum_{i\in\hat{t}} \sum_{j\in\hat{s}} |x_{ij}|^2 = \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \|X|_{\hat{t}\times\hat{s}}\|_F^2,$$

and taking the square root yields the equation. ∎

If we compute low-rank approximations via the singular value decomposition or the rank-revealing QR or LR decomposition, we have the Frobenius norm of the error at our disposal and can apply Lemma 4.45 directly.

If the error results from an orthogonal projection, i.e., if we have

$$G|_{\hat{t}\times\hat{s}} - Q_{ts}Q_{ts}^* G|_{\hat{t}\times\hat{s}}$$

with an isometric matrix Q_{ts}, we can use Lemma 4.28 to obtain

$$\|G|_{\hat{t}\times\hat{s}} - Q_{ts}Q_{ts}^* G|_{\hat{t}\times\hat{s}}\|_F^2 = \|G|_{\hat{t}\times\hat{s}}\|_F^2 - \|Q_{ts}Q_{ts}^* G|_{\hat{t}\times\hat{s}}\|_F^2 = \|G|_{\hat{t}\times\hat{s}}\|_F^2 - \|Q_{ts}^* G|_{\hat{t}\times\hat{s}}\|_F^2.$$

This local error equation can now be combined with Lemma 4.45. If the error is small, we should keep in mind that the right-hand side of this equation may be very susceptible to rounding errors.

If we can write the approximations of submatrices in terms of orthogonal projections with respect to the Frobenius inner product, i.e., if we have a family $(P_{ts})_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}}$ of linear matrix-valued operators

$$P_{ts}\colon \mathbb{K}^{\hat{t}\times\hat{s}} \to \mathbb{K}^{\hat{t}\times\hat{s}} \qquad \text{for all } b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}$$

with

$$P_{ts}^2 = P_{ts}, \quad \langle P_{ts}[Y], Z\rangle_F = \langle Y, P_{ts}[Z]\rangle_F \qquad \text{for all } b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}},\ Y,Z\in\mathbb{K}^{\hat{t}\times\hat{s}},$$

we can use Corollary 3.23 to define a *global* orthogonal projection

$$P\colon \mathbb{K}^{\mathcal{I}\times\mathcal{J}} \to \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$$

via the equations

$$P[Y]|_{\hat{t}\times\hat{s}} := P_{ts}[Y|_{\hat{t}\times\hat{s}}] \qquad \text{for all } b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}},\ Y\in\mathbb{K}^{\mathcal{I}\times\mathcal{J}}.$$

Pythagoras' identity yields $\|Y-P[Y]\|_F^2 = \|Y\|_F^2 - \|P[Y]\|_F^2$. Again, if the error is small, the right-hand side may suffer from rounding errors.

Handling the spectral norm of a matrix is considerably more challenging, since it is defined via a maximum, and the element maximizing the norm may differ from submatrix to submatrix. The estimate (4.8b) yields $\|X\|_2 \le \|X\|_F$, but this may be fairly inaccurate and requires us to have Frobenius norm estimates for all blocks.

If we only have spectral norm estimates, we can still find an upper bound similar to the one provided by Lemma 4.45:

Lemma 4.46 (Global spectral norm) *Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$. We have*

$$\|X\|_2 \leq \left(\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \|X|_{\hat{t}\times\hat{s}}\|_2^2 \right)^{1/2}.$$

Proof. Let $y \in \mathbb{K}^{\mathcal{I}}$ and $z \in \mathbb{K}^{\mathcal{J}}$. Using again Corollary 3.23, we obtain

$$y^* X z = \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} \bar{y}_i x_{ij} z_j = \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \sum_{i\in\hat{t}} \sum_{j\in\hat{s}} \bar{y}_i x_{ij} z_j = \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} y|_{\hat{t}}^* X|_{\hat{t}\times\hat{s}} z|_{\hat{s}}.$$

For every block $b = (t, s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$, we can use the Cauchy-Schwarz inequality and (4.10a) to get

$$|\, y|_{\hat{t}}^* X|_{\hat{t}\times\hat{s}} z|_{\hat{s}} \,| \leq \|y|_{\hat{t}}\|_2 \, \|X|_{\hat{t}\times\hat{s}} z|_{\hat{s}}\|_2 \leq \|y|_{\hat{t}}\|_2 \, \|X|_{\hat{t}\times\hat{s}}\|_2 \, \|z|_{\hat{s}}\|_2.$$

The triangle inequality and the Cauchy-Schwarz inequality applied to the block sum yield

$$\begin{aligned}|y^* X z| &\leq \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} |\, y|_{\hat{t}}^* X|_{\hat{t}\times\hat{s}} z|_{\hat{s}} \,| \leq \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \|y|_{\hat{t}}\|_2 \, \|X|_{\hat{t}\times\hat{s}}\|_2 \, \|z|_{\hat{s}}\|_2 \\ &\leq \left(\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \|X|_{\hat{t}\times\hat{s}}\|_2^2 \right)^{1/2} \left(\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \|y|_{\hat{t}}\|_2^2 \|z|_{\hat{s}}\|_2^2 \right)^{1/2}.\end{aligned}$$

Corollary 3.23 allows us to rewrite the right-hand term as

$$\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \|y|_{\hat{t}}\|_2^2 \|z|_{\hat{s}}\|_2^2 = \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \sum_{i\in\hat{t}} \sum_{j\in\hat{s}} |y_i|^2 |z_j|^2 = \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} |y_i|^2 |z_j|^2 = \|y\|_2^2 \|z\|_2^2,$$

and we conclude

$$|y^* X z| \leq \left(\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \|X|_{\hat{t}\times\hat{s}}\|_2^2 \right)^{1/2} \|y\|_2 \|z\|_2.$$

Now we can apply equation (4.10b) of Lemma 4.24 to complete the proof. ∎

Although the estimate provided by Lemma 4.46 is quite convenient, it is far from optimal: if we consider the identity matrix $X = I$, the spectral norm is equal to one, but the estimate of Lemma 4.46 would be the square root of the number of diagonal blocks, which may be considerably larger.

We can get a far better estimate if we follow the approach of [20, Satz 6.2] and take advantage of the sparsity of the block tree.

Theorem 4.47 (Global spectral norm) *Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be C_{sp}-sparse, let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$.*

If there are families $(\epsilon_{\mathcal{I},\ell})_{\ell=0}^{\infty}$ and $(\epsilon_{\mathcal{J},\ell})_{\ell=}^{\infty}$ in $\mathbb{R}_{\geq 0}$ satisfying

$$\|X|_{\hat{t}\times\hat{s}}\|_2 \leq \epsilon_{\mathcal{I},\mathrm{level}(t)}^{1/2} \, \epsilon_{\mathcal{J},\mathrm{level}(s)}^{1/2} \qquad \textit{for all } b = (t, s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}, \tag{4.17}$$

we have

$$\|X\|_2 \leq C_{\mathrm{sp}} \left(\sum_{\ell=0}^{\infty} \epsilon_{\mathcal{I},\ell}\right)^{1/2} \left(\sum_{\ell=0}^{\infty} \epsilon_{\mathcal{J},\ell}\right)^{1/2}.$$

Proof. Let $(\epsilon_{\mathcal{I},\ell})_{\ell=0}^{\infty}$ and $(\epsilon_{\mathcal{J},\ell})_{\ell=0}^{\infty}$ be families in $\mathbb{R}_{\geq 0}$ satisfying (4.17).

Let $y \in \mathbb{K}^{\mathcal{I}}$ and $z \in \mathbb{K}^{\mathcal{J}}$. As in the previous proof, we use Corollary 3.23, the triangle inequality, the Cauchy-Schwarz inequality and (4.10a) to obtain

$$\begin{aligned} |y^* X z| &= \Big| \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} y|_{\hat{t}}^* \, X|_{\hat{t}\times\hat{s}} \, z|_{\hat{s}} \Big| \leq \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} |\, y|_{\hat{t}}^* \, X|_{\hat{t}\times\hat{s}} \, z|_{\hat{s}} \,| \\ &\leq \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \|y|_{\hat{t}}\|_2 \, \|X|_{\hat{t}\times s}\|_2 \, \|z|_{\hat{s}}\|_2. \end{aligned}$$

Now we use (4.17) and apply the Cauchy-Schwarz inequality to the sum to get

$$\begin{aligned} |y^* X z| &\leq \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \|y|_{\hat{t}}\|_2 \, \epsilon_{\mathcal{I},\mathrm{level}(t)}^{1/2} \epsilon_{\mathcal{J},\mathrm{level}(s)}^{1/2} \|z|_{\hat{s}}\|_2 \\ &\leq \left(\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \|y|_{\hat{t}}\|_2^2 \, \epsilon_{\mathcal{I},\mathrm{level}(t)} \right)^{1/2} \left(\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \|z|_{\hat{s}}\|_2^2 \, \epsilon_{\mathcal{J},\mathrm{level}(s)} \right)^{1/2}. \end{aligned}$$

For the first term, we can take advantage of the block tree's sparsity and Lemma 3.18 to get

$$\begin{aligned} \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \epsilon_{\mathcal{I},\mathrm{level}(t)} \, \|y|_{\hat{t}}\|_2^2 &= \sum_{t\in\mathcal{T}_{\mathcal{I}}} \sum_{s\in\mathrm{row}(t)} \epsilon_{\mathcal{I},\mathrm{level}(t)} \, \|y|_{\hat{t}}\|_2^2 \leq C_{\mathrm{sp}} \sum_{t\in\mathcal{T}_{\mathcal{I}}} \epsilon_{\mathcal{I},\mathrm{level}(t)} \, \|y|_{\hat{t}}\|_2^2 \\ &= C_{\mathrm{sp}} \sum_{\ell=0}^{\infty} \sum_{\substack{t\in\mathcal{T}_{\mathcal{I}} \\ \mathrm{level}(t)=\ell}} \epsilon_{\mathcal{I},\ell} \, \|y|_{\hat{t}}\|_2^2 = C_{\mathrm{sp}} \sum_{\ell=0}^{\infty} \epsilon_{\mathcal{I},\ell} \sum_{\substack{t\in\mathcal{T}_{\mathcal{I}} \\ \mathrm{level}(t)=\ell}} \sum_{i\in\hat{t}} |y_i|^2 \\ &\leq C_{\mathrm{sp}} \sum_{\ell=0}^{\infty} \epsilon_{\mathcal{I},\ell} \sum_{i\in\mathcal{I}} |y_i|^2 \leq C_{\mathrm{sp}} \sum_{\ell=0}^{\infty} \epsilon_{\mathcal{I},\ell} \, \|y\|_2^2. \end{aligned}$$

By the same arguments, we also find

$$\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \epsilon_{\mathcal{J},\mathrm{level}(s)} \, \|z|_{\hat{s}}\|_2^2 \leq C_{\mathrm{sp}} \sum_{\ell=0}^{\infty} \epsilon_{\mathcal{J},\ell} \|z\|_2^2,$$

and combining both estimates yields

$$|y^* X z| \leq C_{\mathrm{sp}} \left(\sum_{\ell=0}^{\infty} \epsilon_{\mathcal{I},\ell}\right)^{1/2} \left(\sum_{\ell=0}^{\infty} \epsilon_{\mathcal{J},\ell}\right)^{1/2} \|y\|_2 \, \|z\|_2,$$

and equation (4.10b) of Lemma 4.24 can again be used to complete the proof. ■

If we have a *level-consistent* block tree, i.e., if the levels of blocks and their row and column clusters coincide, this result can be made a little more accessible.

Corollary 4.48 (Global spectral norm) *Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be C_{sp}-sparse with*

$$\mathrm{level}(b) = \mathrm{level}(t) = \mathrm{level}(s) \qquad \text{for all } b = (t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}. \tag{4.18}$$

Let $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$. We have

$$\|X\|_2 \leq C_{\mathrm{sp}} \sum_{\ell=0}^{\infty} \max\{\|X|_{\hat{t}\times\hat{s}}\|_2 \ : \ b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}},\ \mathrm{level}(b) = \ell\}.$$

Proof. We simply let

$$\epsilon_{\mathcal{I},\ell} = \epsilon_{\mathcal{J},\ell} := \max\{\|X|_{\hat{t}\times\hat{s}}\|_2 \ : \ b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}},\ \mathrm{level}(b) = \ell\} \qquad \text{for all } \ell \in \mathbb{N}_0.$$

Due to (4.18), the condition (4.17) of Theorem 4.47 is fulfilled and we get

$$\|X\|_2 \leq C_{\mathrm{sp}} \sum_{\ell=0}^{\infty} \epsilon_{\mathcal{I},\ell}.$$

This is already the desired result. ■

5 Arithmetic operations

The discretization of integral or partial differential equations typically leads to large ill-conditioned linear systems that require efficient solvers. Hierarchical matrices offer an elegant approach to this challenge: we can formulate efficient algorithms that approximate the inverse or the factorization of hierarchical matrices by simply replacing standard arithmetic operations by *truncated* operations that reduce the rank of suitable submatrices after each step. This approach ensures that all intermediate results can be handled efficiently.

5.1 Matrix-vector multiplication

Let $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ be an $\mathcal{H}$-matrix corresponding to an admissible block tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ for cluster trees $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$, and let (A, B, N) be an $\mathcal{H}$-matrix representation of G.

We are interested in evaluating the matrix-vector product Gy for a vector $y \in \mathbb{K}^{\mathcal{J}}$. Since we will frequently have to apply this operation to multiple vectors y at once, we combine multiple vectors into the columns of a matrix $Y \in \mathbb{K}^{\mathcal{J}\times\mathcal{M}}$ and consider the computation of $GY \in \mathbb{K}^{\mathcal{I}\times\mathcal{M}}$. Since we will frequently require the restriction to a subset of the rows of the matrix, we introduce the notation

$$X|_{\hat{t}'} := X|_{\hat{t}'\times\mathcal{M}} \qquad \text{for all } X \in \mathbb{K}^{\hat{t}\times\mathcal{M}},\ \hat{t}' \subseteq \hat{t}, \tag{5.1}$$

where $\mathcal{M}$ is an arbitrary finite index set. In order to obtain a flexible algorithm, we focus on the update operation

$$X \leftarrow X + \alpha G|_{\hat{t}\times\hat{s}} Y \tag{5.2}$$

for a block $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with a result matrix $X \in \mathbb{K}^{\hat{t}\times\mathcal{M}}$, an input matrix $Y \in \mathbb{K}^{\hat{s}\times\mathcal{M}}$, and a scaling factor $\alpha \in \mathbb{K}$.

In order to find an efficient algorithm for performing this operation, we distinguish three types of blocks: if $\operatorname{chil}(b) \neq \emptyset$, the block can be *subdivided* into blocks corresponding to its children. If $\operatorname{chil}(b) = \emptyset$, the block can be either an *admissible* or an *inadmissible* leaf of the block tree.

In the first case, i.e., if $\operatorname{chil}(b) \neq \emptyset$, we have

$$\operatorname{chil}(b) = \operatorname{chil}^+(t) \times \operatorname{chil}^+(s).$$

By Definition 3.20, we obtain

$$(G|_{\hat{t}\times\hat{s}}Y)|_{\hat{t}'} = G|_{\hat{t}'\times\hat{s}}Y = \sum_{s'\in\operatorname{chil}^+(s)} G|_{\hat{t}'\times\hat{s}'}Y|_{\hat{s}'} \qquad \text{for all } t' \in \operatorname{chil}^+(t).$$

```
procedure addeval_hmatrix ($\alpha$, $G$, $b = (t, s)$, $Y$, var $X$);
  if $b \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$ then
    $X \leftarrow X + \alpha N_b Y$
  else if $b \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ then
    $\widehat{Y} \leftarrow \alpha B_b^* Y$;
    $X \leftarrow X + A_b \widehat{Y}$
  else for $b' = (t', s') \in \operatorname{chil}(b)$ do
    addeval_hmatrix ($\alpha$, $G$, $b'$, $Y|_{\hat{s}'}$, $X|_{\hat{t}'}$)
end
```

Figure 5.1: Matrix-vector multiplication $X \leftarrow X + \alpha G|_{\hat{t}\times\hat{s}} Y$

Lemma 3.21 states that the sets $\{\hat{t}' \ : \ t' \in \operatorname{chil}^+(t)\}$ are a disjoint partition of $\hat{t}$, so we may conclude that (5.2) is equivalent with the operations

$$X|_{\hat{t}'} \leftarrow X|_{\hat{t}'} + \alpha\, G|_{\hat{t}'\times\hat{s}'} Y|_{\hat{s}'} \qquad \text{for all } b' = (t', s') \in \operatorname{chil}(b),$$

i.e., it suffices to perform updates for all children of b. We can repeat this procedure recursively until we arrive of the leaves of the block tree.

If $b = (t, s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$ is an admissible leaf, the $\mathcal{H}$-matrix representation yields $A_b \in \mathbb{K}^{\hat{t}\times k}$ and $B_b \in \mathbb{K}^{\hat{s}\times k}$ with $G|_{\hat{t}\times\hat{s}} = A_b B_b^*$. Due to

$$G|_{\hat{t}\times\hat{s}} Y = A_b B_b^* Y,$$

we can compute

$$\widehat{Y} := \alpha B_b^* Y, \qquad X + \alpha G|_{\hat{t}\times\hat{s}} \widehat{Y} = X + A_b \widehat{Y}.$$

If $b = (t, s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$ is an inadmissible leaf, the $\mathcal{H}$-matrix representation gives us $N_b \in \mathbb{K}^{\hat{t}\times\hat{s}}$ with $G|_{\hat{t}\times\hat{s}} = N_b$, and the product

$$X + \alpha G|_{\hat{t}\times\hat{s}} Y = X + \alpha N_b Y$$

can be evaluated directly. The resulting algorithm is summarized in Figure 5.1.

Occasionally we also require an algorithm for computing matrix-vector products with the adjoint matrix, i.e.,

$$X \leftarrow X + \alpha G^* Y \tag{5.3}$$

with $X \in \mathbb{K}^{\hat{s}\times\ell}$ and $Y \in \mathbb{K}^{\hat{t}\times\ell}$. We can treat the admissible leaves $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ by computing

$$\widehat{Y} := \alpha A_b^* Y, \qquad G|^*_{\hat{t}\times\hat{s}} Y = B_b \widehat{Y},$$

while the inadmissible leaves $b = (t, s) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$ can be handled directly.

Recursively following the structure of the block tree leads to the algorithm summarized in Figure 5.2.

```
procedure addevaltrans_hmatrix ($\alpha$, $G$, $b = (t,s)$, $Y$, var $X$);
  if $b \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$ then
    $X \leftarrow X + \alpha N_b^* Y$
  else if $b \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ then
    $\widehat{Y} \leftarrow \alpha A_b^* Y$;
    $X \leftarrow X + B_b \widehat{Y}$
  else for $b' = (t', s') \in \operatorname{chil}(b)$ do
    addevaltrans_hmatrix ($\alpha$, $G$, $b'$, $Y|_{\hat{t}'}$, $X|_{\hat{s}'}$)
end
```

Figure 5.2: Adjoint matrix-vector multiplication $X \leftarrow X + \alpha G|_{\hat{t}\times\hat{s}}^* Y$

Remark 5.1 (Auxiliary vector) *It is possible to avoid the auxiliary vectors $\widehat{Y}$ by treating $A_b B_b^*$ as a sequence of k rank-one updates. As in Theorem 4.6, we let*

$$A_b = \begin{pmatrix} a_1 & a_2 & \dots & a_k \end{pmatrix}, \qquad B_b = \begin{pmatrix} b_1 & b_2 & \dots & b_k \end{pmatrix}$$

and find

$$A_b B_b^* Y = \sum_{\nu=1}^{k} a_\nu b_\nu^* Y.$$

The individual rank-one updates require us to store only the intermediate results $b_\nu^* Y$ instead of the entire vector $\widehat{Y}$, and these intermediate results can be computed and used component by component, reducing the auxiliary storage requirements to $\mathcal{O}(1)$.

5.2 Complexity of the matrix-vector multiplication

Now we consider the amount of computational work required to perform a matrix-vector multiplication by the algorithms `addeval_hmatrix` and `addevaltrans_hmatrix` presented in Figure 5.1 and Figure 5.2. Since most of the work is done in the leaves of the block tree, it is convenient to use their ranks to bound the overall work.

Lemma 5.2 (Maximal rank) *Let $r_\mathcal{I}$ and $r_\mathcal{J}$ denote the resolutions of the cluster trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$, and let k denote the local rank of G. The* maximal rank *of G is given by*

$$\hat{k} := \max\{k, r_\mathcal{I}, r_\mathcal{J}\}. \tag{5.4}$$

We have

$$\operatorname{rank}(G|_{\hat{t}\times\hat{s}}) \le \hat{k} \qquad \text{for all leaves } b = (t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}.$$

Proof. Let $b = (t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$. If b is an admissible leaf, Definition 3.26 implies that the rank of $G|_{\hat{t}\times\hat{s}}$ is bounded by k. If b is an inadmissible leaf, Definition 3.28 implies that t or s has to be a leaf, so the rank of $G|_{\hat{t}\times\hat{s}}$ is bounded by $r_\mathcal{I}$ or $r_\mathcal{J}$. ∎

Since the algorithms only consider blocks and their descendants, the computational work should only depend on these blocks. To express this property, we introduce *subtrees* of cluster and block trees.

Lemma 5.3 (Subtree) *Let $\mathcal{T} = (V, r, E)$ be a tree, and let $r' \in V$. Let $V' \subseteq V$ be the minimal subset satisfying*

- *$r' \in V'$ and*
- *$\operatorname{chil}(v) \subseteq V'$ for all $v \in V'$.*

Let $E' := E \cap (V' \times V')$. Then $\mathcal{T}' := (V', r', E')$ is a tree, and we call it the subtree *of $\mathcal{T}$ for the root r'.*

We have $V' = \operatorname{desc}(r')$, i.e., the nodes of the subtree are the descendants of its root r'.

Proof. Let $v \in V'$. Since $\mathcal{T}$ is a tree, there is exactly one sequence $v_0, \ldots, v_\ell \in V$, $\ell \in \mathbb{N}_0$, such that

$$(v_{i-1}, v_i) \in E \qquad\qquad \text{for all } i \in [1 : \ell],$$

and $v_0 = r$, $v_\ell = v$. Let $\lambda := \min\{i \in [0 : \ell] \ : \ v_i \in V'\}$. Due to the minimality of λ, v_λ is not a child of an element in V'. Due to the minimality of V', it therefore has to be r'. We conclude that $v_\lambda, \ldots, v_\ell$ is a sequence in V' connecting $r' = v_\lambda$ to $v = v_\ell$.

If we have two sequences connecting r' to v, we can extend them to sequences connecting r to v. Since these sequences are unique by definition of the tree $\mathcal{T}$, so are the sequences in the subtree $\mathcal{T}'$. We may conclude that $\mathcal{T}'$ is indeed a tree.

The definition of V' compared to Definition 3.17 immediately implies $\operatorname{desc}(r') \subseteq V'$. On the other hand, we have already proven that every $v \in v'$ is a descendant of r'. ■

Notation 5.4 (Subtrees) *We introduce the following abbreviations:*

For every $t \in \mathcal{T}_{\mathcal{I}}$, we denote the subtree of $\mathcal{T}_{\mathcal{I}}$ with the root t by $\mathcal{T}_t$.

For every $s \in \mathcal{T}_{\mathcal{J}}$, we denote the subtree of $\mathcal{T}_{\mathcal{J}}$ with the root s by $\mathcal{T}_s$.

For every $b \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, we denote the subtree of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with the root b by $\mathcal{T}_b$.

The cardinality of the set of nodes of a tree $\mathcal{T} = (V, r, E)$ is denoted by $|\mathcal{T}| := |V|$.

In order to obtain a bound for the number of operations required by the matrix-vector multiplication, we use three steps: first we derive a recursive estimate that closely follows the structure of the algorithm. This estimate can be used to express the work as a sum of the numbers of operations in all blocks. Finally we use properties like sparsity to translate the second estimate into an explicit upper bound.

This approach provides us with auxiliary results that are very useful for more sophisticated algorithms like the $\mathcal{H}$-matrix inversion or the $\mathcal{H}$-matrix multiplication that use the matrix-vector multiplication of submatrices in intermediate steps.

Lemma 5.5 (Computational work) *We define*

$$W_{mv}(t,s,\mathcal{M}) := \begin{cases} 2\hat{k}\,|\mathcal{M}|\,(|\hat{t}| + |\hat{s}|) & \text{if } (t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}, \\ \sum_{(t',s')\in\operatorname{chil}(t,s)} W_{mv}(t',s',\mathcal{M}) & \text{otherwise} \end{cases}$$

for all $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ and finite sets $\mathcal{M}$.

If we call the algorithm **addeval_hmatrix** *for a block $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with matrices $X \in \mathbb{K}^{\hat{t}\times\mathcal{M}}$ and $Y \in \mathbb{K}^{\hat{s}\times\mathcal{M}}$ or the algorithm* **addevaltrans_hmatrix** *for a block $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with matrices $X \in \mathbb{K}^{\hat{s}\times\mathcal{M}}$ and $Y \in \mathbb{K}^{\hat{t}\times\mathcal{M}}$, it performs not more than $W_{mv}(t,s,\mathcal{M})$ operations.*

Proof. We consider only `addeval_hmatrix`, since `addevaltrans_hmatrix` performs the same number of operations, just exchanging the roles of A_b and B_b for admissible leaves and using the adjoint in inadmissible leaves.

By induction on $|\mathcal{T}_b| = |\operatorname{desc}(b)|$.

Let $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with $|\mathcal{T}_b| = 1$. Then we have $\operatorname{chil}(b) = \emptyset$, i.e., $b \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$.

If b is admissible, the algorithm computes

$$\widehat{Y} \leftarrow \alpha B_b^* Y, \qquad\qquad X \leftarrow X + A_b\widehat{Y}.$$

The multiplication with B_b^* takes $k\,|\mathcal{M}|\,(2\,|\hat{s}| - 1)$ operations, scaling the result takes $k\,|\mathcal{M}|$ operations, the multiplication with A_b takes $|\hat{t}|\,|\mathcal{M}|\,(2k-1)$ operations, and adding the result to $X|_{\hat{t}}$ takes $|\hat{t}|\,|\mathcal{M}|$. In total, the algorithm `addeval_hmatrix` requires not more than

$$2k\,|\mathcal{M}|\,(|\hat{t}| + |\hat{s}|) \leq 2\hat{k}\,|\mathcal{M}|\,(|\hat{t}| + |\hat{s}|) \text{ operations.}$$

If b is inadmissible, we have to distinguish to cases: if $|\hat{t}| \leq |\hat{s}|$, we compute

$$X + \alpha G|_{\hat{t}\times\hat{s}} Y = X + \alpha(N_b Y)$$

using $|\hat{t}|\,|\mathcal{M}|\,(2|\hat{s}| - 1)$ operations for the matrix multiplication, $|\hat{t}|\,|\mathcal{M}|$ operations for scaling with α, and finally $|\hat{t}|\,|\mathcal{M}|$ operations to add the result to X. Due to $|\hat{t}| \leq |\hat{s}|$, the total number of operations is bounded by

$$2|\hat{t}|\,|\mathcal{M}|\,|\hat{s}| + |\mathcal{M}|\,|\hat{t}| \leq 2|\hat{t}|\,|\mathcal{M}|\,|\hat{s}| + |\mathcal{M}|\,\min\{|\hat{t}|, |\hat{s}|\}.$$

If, on the other hand $|\hat{t}| > |\hat{s}|$, we compute

$$X + \alpha G|_{\hat{t}\times\hat{s}} Y = X + N_b(\alpha Y).$$

Scaling Y takes $|\mathcal{M}|\,|\hat{s}|$ operations, multiplying by N_b takes $|\hat{t}|\,|\mathcal{M}|\,(2|\hat{s}| - 1)$, and adding to $X|_{\hat{t}}$ takes $|\hat{t}|\,|\mathcal{M}|$. Due to $|\hat{t}| > |\hat{s}|$, the total number of operations is bounded by

$$2|\hat{t}|\,|\mathcal{M}|\,|\hat{s}| + |\mathcal{M}|\,|\hat{s}| \leq 2|\hat{t}|\,|\mathcal{M}|\,|\hat{s}| + |\mathcal{M}|\,\min\{|\hat{t}|, |\hat{s}|\}.$$

Since $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is an admissible block tree, either t or s has to be a leaf. In the first case, we obtain the bound

$$2|\hat{t}|\,|\mathcal{M}|\,|\hat{s}| + |\mathcal{M}|\,\min\{|\hat{t}|, |\hat{s}|\} \leq 2r_{\mathcal{I}}\,|\mathcal{M}|\,|\hat{s}| + |\mathcal{M}|\,|\hat{t}| \leq 2\hat{k}\,|\mathcal{M}|(|\hat{t}| + |\hat{s}|).$$

In the second case, we find

$$2|\hat{s}|\,|\mathcal{M}|\,|\hat{t}| + |\mathcal{M}|\,\min\{|\hat{t}|,|\hat{s}|\} \le 2r_{\mathcal{J}}|\mathcal{M}|\,|\hat{t}| + |\mathcal{M}|\,|\hat{s}| \le 2\hat{k}\,|\mathcal{M}|\,(|\hat{t}|+|\hat{s}|).$$

Combining our three estimates, we conclude that the algorithm requires not more than

$$2\hat{k}\,|\mathcal{M}|(|\hat{t}|+|\hat{s}|) = W_{\mathrm{mv}}(t,s,\mathcal{M}) \text{ operations}$$

if $b=(t,s)$ is a leaf block.

Let now $m \in \mathbb{N}$ be given such that our claim holds for all $b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with $|\mathcal{T}_b| \le m$.

Let $b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with $|\mathcal{T}_b| = m+1$. Then we have $\operatorname{chil}(b)\neq\emptyset$ and the algorithm `addeval_hmatrix` calls itself recursively for all $b'=(t',s')\in\operatorname{chil}(b)$. Due to $|\mathcal{T}_b|\le m$, we can apply the induction assumption to find that each of these recursive calls requires not more than $W_{\mathrm{mv}}(t',s',\mathcal{M})$ operations, so the total is bounded by

$$\sum_{(t',s')\in\operatorname{chil}(t,s)} W_{\mathrm{mv}}(t',s',\mathcal{M}) = W_{\mathrm{mv}}(t,s,\mathcal{M}).$$

The induction is complete. ∎

Theorem 5.6 (Complexity) *Let $\mathcal{M}$ be a finite set. Calling `addeval_hmatrix` or `addevaltrans_hmatrix` with $b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ requires not more than*

$$W_{mv}(t,s,\mathcal{M}) \le 2\hat{k}\,|\mathcal{M}| \sum_{b'=(t',s')\in\mathcal{T}_b} (|\hat{t}'|+|\hat{s}'|) \text{ operations.} \tag{5.5a}$$

If $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is C_{sp}-sparse, we find

$$W_{mv}(t,s) \le 2C_{\mathrm{sp}}\hat{k}\,|\mathcal{M}|(p_{\mathcal{I}\times\mathcal{J}}+1)(|\hat{t}|+|\hat{s}|). \tag{5.5b}$$

Proof. We prove (5.5a) by induction on $|\mathcal{T}_b| = |\operatorname{desc}(b)|$.

Let $b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with $|\mathcal{T}_b|=1$. Then b is a leaf and the estimate follows directly from Lemma 5.5.

Let now $m\in\mathbb{N}$ be given such that (5.5a) holds for all $b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with $|\mathcal{T}_b|\le m$.

Let $b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with $|\mathcal{T}_b|=m+1$. According to Lemma 5.5, we have

$$W_{\mathrm{mv}}(t,s,\mathcal{M}) = \sum_{(t',s')\in\operatorname{chil}(t,s)} W_{\mathrm{mv}}(t',s',\mathcal{M}),$$

and due to $|\mathcal{T}_{b'}|\le m$ for all $b'=(t',s')\in\operatorname{chil}(b)$, we can apply the induction assumption to obtain

$$\begin{aligned} W_{\mathrm{mv}}(t,s,\mathcal{M}) &\le \sum_{b'=\operatorname{chil}(b)} \sum_{b''=(t'',s'')\in\mathcal{T}_{b'}} 2\hat{k}\,|\mathcal{M}|\,(|\hat{t}''|+|\hat{s}''|) \\ &\le 2\hat{k}\,|\mathcal{M}| \sum_{b'\in\mathcal{T}_b} (|\hat{t}'|+|\hat{s}'|). \end{aligned}$$

Combining (5.5a) with Lemma 3.34 yields (5.5b). ∎

5.3 Truncation

In order to approximate the results of arithmetic operations, we require an algorithm that keeps the rank of the resulting matrices as low as possible. The truncation strategy based on the singular value decomposition discussed in Section 4.2 yields the best possible result (cf. Theorem 4.29), but computing the decomposition "from scratch" would lead to a very high computational complexity. Fortunately, we can arrange a number of important arithmetic operations in a way that yields $\mathcal{R}(k)$-matrix representations of submatrices, where k may be higher than necessary, but not too high. Taking advantage of the factorized form, we can obtain the singular value decomposition by an efficient algorithm.

We assume that a matrix $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ is given in $\mathcal{R}(k)$-matrix representation, i.e., that there are matrices $A \in \mathbb{K}^{\mathcal{I}\times k}$ and $B \in \mathbb{K}^{\mathcal{J}\times k}$ such that

$$X = AB^*$$

holds. Instead of multiplying A and B^* to form X, we first compute a thin QR factorization of A, i.e., we find an isometric matrix $Q \in \mathbb{K}^{\mathcal{I}\times k}$ and an upper (according to an arbitrary ordering of indices) triangular matrix $R \in \mathbb{K}^{k\times k}$ satisfying

$$A = QR.$$

We find

$$X = AB^* = QRB^*$$

and introducing

$$\widehat{X} := RB^* \in \mathbb{K}^{k\times\mathcal{J}}$$

this equation takes the form

$$X = Q\widehat{X}.$$

Assuming that k is not too large, we can afford to compute its singular value decomposition

$$\widehat{X} = \widehat{U}\Sigma V^*$$

and obtain

$$X = Q\widehat{U}\Sigma V^* = U\Sigma V^*$$

with $U := Q\widehat{U}$.

We can, of course, interchange the roles of A and B: we can also compute the QR factorization $B = QR$ with an isometric matrix $Q \in \mathbb{K}^{\mathcal{J}\times k}$ and an upper triangular matrix $R \in \mathbb{K}^{k\times k}$, set up $\widehat{X} := AR^*$, compute the SVD $\widehat{X} = U\Sigma\widehat{V}^*$, and obtain

$$X = \widehat{X}Q^* = U\Sigma\widehat{V}^*Q^* = U\Sigma V^*$$

with $V := Q\widehat{V}$. Depending on the implementation of the QR factorization and the singular value decomposition and on the cardinalities $|\mathcal{I}|$ and $|\mathcal{J}|$, one of the two versions may be more efficient than the other.

```
procedure trunc_rkmatrix ($\epsilon$, var $A$, $B$);
  Compute QR factorization $A = QR$ with $Q \in \mathbb{K}^{\mathcal{I}\times k}$, $R \in \mathbb{K}^{k\times k}$;
  $\widehat{X} \leftarrow RB^*$;
  Compute singular value decomposition $\widehat{X} = \widehat{U}\Sigma V^*$;
  $U \leftarrow Q\widehat{U}$;
  Choose rank $\tilde{k} \in [0 : k]$;
  $A \leftarrow U|_{\mathcal{I}\times\tilde{k}}\Sigma|_{\tilde{k}\times\tilde{k}}$;    $B \leftarrow V|_{\mathcal{J}\times\tilde{k}}$
end
```

Figure 5.3: $\mathcal{R}(k)$-matrix truncation, X is overwritten by an approximation.

Given the SVD, computing a rank-$\tilde{k}$ approximation for a given $\tilde{k} \in [0 : k]$ consists of merely copying the appropriate columns of U and V and scaling one of them by the singular values. The resulting algorithm is presented in Figure 5.3.

Assumption 5.7 (QR and SVD) *We assume that there is a constant C_{qr} such that the QR factorization of a matrix $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ can be computed in not more than*

$$C_{qr}|\mathcal{I}|\,|\mathcal{J}|\,\min\{|\mathcal{I}|, |\mathcal{J}|\} \text{ operations}$$

and that applying Q or Q^* to a matrix $Y \in \mathbb{K}^{\mathcal{I}\times\mathcal{K}}$ takes not more than

$$C_{qr}|\mathcal{I}|\,|\mathcal{K}|\,\min\{|\mathcal{I}|, |\mathcal{J}|\} \text{ operations.}$$

We also assume that there is a constant C_{svd} such that the singular value decomposition of X can be computed in not more than

$$C_{svd}|\mathcal{I}|\,|\mathcal{J}|\,\min\{|\mathcal{I}|, |\mathcal{J}|\} \text{ operations.}$$

In practice, we of course only refer to computations with error tolerances reasonably close to machine accuracy.

Lemma 5.8 (Complexity of $\mathcal{R}(k)$-SVDs) *Our algorithm computes the singular value decomposition of a matrix $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ given in $\mathcal{R}(k)$-matrix representation in not more than*

$$C_{rksvd}k^2(|\mathcal{I}| + |\mathcal{J}|) \text{ operations,}$$

where $C_{rksvd} := \max\{2C_{qr}, C_{svd} + 2\}$.

Proof. By Assumption 5.7, the construction of the QR factorization takes not more than $C_{\mathrm{qr}}k^2\,|\mathcal{I}|$ operations, the product $\widehat{X} = RB^*$ can be computed in not more than $2k^2\,|\mathcal{J}|$ operations, its singular value decomposition takes not more than $C_{\mathrm{svd}}k^2\,|\mathcal{J}|$ operations, and finally the product $U = Q\widehat{U}$ can be obtained in $C_{\mathrm{qr}}k^2\,|\mathcal{I}|$ operations.

Adding the contributions yields

$$2C_{\mathrm{qr}}k^2\,|\mathcal{I}| + (C_{\mathrm{svd}} + 2)k^2\,|\mathcal{J}| \le C_{\mathrm{rksvd}}k^2(|\mathcal{I}| + |\mathcal{J}|).$$

■

```
procedure add_rkmatrix ($\epsilon$, $\alpha$, $A$, $B$, var $A_X$, $B_X$);
  $A_X \leftarrow \begin{pmatrix} A_X & A \end{pmatrix}$;   $B_X \leftarrow \begin{pmatrix} B_X & \bar{\alpha} B \end{pmatrix}$
  trunc_rkmatrix($\epsilon$, $A_X$, $B_X$)
end
```

Figure 5.4: $\mathcal{R}(k)$-matrix addition, X is overwritten by an approximation of $X + \alpha Y$.

Corollary 5.9 (Complexity of $\mathcal{R}(k)$-truncations) *For any $\tilde{k} \in [0 : k]$, our algorithm `trunc_rkmatrix` computes a rank-$\tilde{k}$ approximation of X in not more than*

$$C_{tr} k^2 (|\mathcal{I}| + |\mathcal{J}|) \text{ operations,}$$

where $C_{tr} := C_{rksvd} + 1$.

Proof. We compute the $\mathcal{R}(k)$-SVD and let

$$\widetilde{A} := U|_{\mathcal{I} \times \tilde{k}} \Sigma|_{\tilde{k} \times \tilde{k}}, \qquad \widetilde{B} := V|_{\mathcal{J} \times \tilde{k}}.$$

The multiplication by the diagonal matrix requires not more than

$$\tilde{k}\, |\mathcal{I}| \le k^2\, |\mathcal{I}| \text{ operations,}$$

and adding the estimate of Lemma 5.8 yields the required bound. ∎

5.4 Low-rank updates

We consider the addition of two matrices $X, Y \in \mathbb{K}^{\mathcal{I} \times \mathcal{J}}$, i.e., the computation of $Z = X + \alpha Y$ with $\alpha \in \mathbb{K}$.

If X and Y are given in $\mathcal{R}(k)$-matrix representation, i.e., if there are matrices $A_X, A_Y \in \mathbb{K}^{\mathcal{I} \times k}$ and $B_X, B_Y \in \mathbb{K}^{\mathcal{J} \times k}$ such that

$$X = A_X B_X^*, \qquad Y = A_Y B_Y^*,$$

we immediately obtain an $\mathcal{R}(2k)$-matrix representation

$$\begin{aligned} X + \alpha Y &= A_X B_X^* + \alpha A_Y B_Y^* = \begin{pmatrix} A_X & A_Y \end{pmatrix} \begin{pmatrix} B_X^* \\ \alpha B_Y^* \end{pmatrix} \\ &= \underbrace{\begin{pmatrix} A_X & A_Y \end{pmatrix}}_{=:A_Z} \underbrace{\begin{pmatrix} B_X & \bar{\alpha} B_Y \end{pmatrix}}_{=:B_Z}{}^* = A_Z B_Z^* \end{aligned}$$

of the sum $X + Y$ and can apply the truncation algorithm to compute an approximation of reduced rank.

Corollary 5.10 (Complexity of $\mathcal{R}(k)$-addition) *Let $X, Y \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ be given in $\mathcal{R}(k)$-matrix representation $X = A_X B_X^*$ and $Y = A_Y B_Y^*$. For any $\tilde{k} \in [0 : 2k]$, the function* ***add_rkmatrix*** *computes a rank-$\tilde{k}$ approximation of $X + \alpha Y$ in not more than*

$$C_{add} k^2(|\mathcal{I}| + |\mathcal{J}|) \text{ operations,}$$

where $C_{add} := 4C_{rksvd} + 2$.

Proof. Constructing A_Z and B_Z requires $k\,|\mathcal{J}|$ multiplications for scaling B_Y. According to Lemma 5.8, we can obtain the singular value decomposition $U\Sigma V^* = A_Z B_Z^*$ in not more than

$$C_{\mathrm{rksvd}}(2k)^2(|\mathcal{I}| + |\mathcal{J}|) = 4C_{\mathrm{rksvd}}k^2(|\mathcal{I}| + |\mathcal{J}|)$$

operations. Finally multiplying $U|_{\mathcal{I}\times\tilde{k}}$ and $\Sigma|_{\tilde{k}\times\tilde{k}}$ takes not more than $2k\,|\mathcal{I}|$ multiplications, since Σ is a diagonal matrix and $\tilde{k} \le 2k$.

The total number of operations is bounded by

$$\begin{aligned} k\,|\mathcal{J}| + 4C_{\mathrm{rksvd}}k^2(|\mathcal{I}| + |\mathcal{J}|) + 2k\,|\mathcal{I}| &\le 4C_{\mathrm{rksvd}}k^2(|\mathcal{I}| + |\mathcal{J}|) + 2k^2(|\mathcal{I}| + |\mathcal{J}|) \\ &= C_{\mathrm{add}}k^2(|\mathcal{I}| + |\mathcal{J}|). \end{aligned}$$

■

In order to approximate the matrix-matrix multiplication, we have to be able to add a low-rank matrix to a hierarchical matrix, i.e., we require an efficient algorithm for performing *low-rank updates* to hierarchical matrices. The restriction of a low-rank matrix $Y \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ to a block $b = (t, s)$ is again of low rank: if $Y = AB^*$ is an $\mathcal{R}(k)$-matrix representation, we have

$$Y|_{\hat{t}\times\hat{s}} = (AB^*)|_{\hat{t}\times\hat{s}} = A|_{\hat{t}\times k} B|^*_{\hat{s}\times k},$$

so an $\mathcal{R}(k)$-matrix representation of $Y|_{\hat{t}\times\hat{s}}$ is readily available to us, and we can split Y into submatrices that fit the block structure of a hierarchical matrix X.

As in the case of the matrix-vector multiplication algorithm, we can use a recursive strategy to distribute the low-rank matrix Y among all leaves of a hierarchical matrix X. For inadmissible leaves, we simply multiply A and B^* and add the result to the corresponding nearfield matrix. For admissible leaves, we use the function `add_rkmatrix`. The resulting algorithm is given in Figure 5.5.

Lemma 5.11 (Complexity of low-rank updates) *Let $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$. A call to the function* ***add_rkmatrix_hmatrix*** *for this block requires not more than*

$$W_{up}(t, s) := C_{up} k\,\hat{k} \sum_{b'=(t',s')\in\mathcal{T}_b} |\hat{t}'| + |\hat{s}'|$$

operations, where $C_{up} := \max\{2, C_{add}\}$.

If $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is C_{sp}-sparse, we find

$$W_{up}(t, s) \le C_{up} C_{\mathrm{sp}} \hat{k}^2 (p_{\mathcal{I}\times\mathcal{J}} + 1)(|\hat{t}| + |\hat{s}|).$$

```
procedure add_rkmatrix_hmatrix ($b = (t,s)$, $\epsilon$, $\alpha$, $A$, $B$, var $X$);
  { $\mathcal{H}$-matrix representation $(A_X, B_X, N_X)$ of $X$ }
  if $b \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$ then
    $N_{X,b} \leftarrow N_{X,b} + \alpha A B^*$
  else if $b \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ then
    add_rkmatrix($\epsilon$, $\alpha$, $A$, $B$, $A_{X,b}$, $B_{X,b}$)
  else for all $b' = (t', s') \in \mathrm{chil}(b)$ do begin
    $\widehat{A} \leftarrow A|_{\hat{t}'\times k}$;   $\widehat{B} \leftarrow B|_{\hat{s}'\times k}$;
    add_rkmatrix_hmatrix($b'$, $\epsilon$, $\alpha$, $\widehat{A}$, $\widehat{B}$, $X$)
  end
end
```

Figure 5.5: $\mathcal{R}(k)$-matrix update, X is overwritten by an approximation of $X + \alpha Y$.

Proof. We prove the estimate by induction on $|\mathcal{T}_b|$, i.e., the cardinality of the subtree with root $b = (t,s)$.

Let $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with $|\mathcal{T}_b = 1|$. Then b is be a leaf of the block tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$.

If b is an inadmissible leaf, we add $\alpha A_Y B_Y^*$ to $N_{X,b}$. If $|\hat{t}| \leq |\hat{s}|$, we first apply the scaling factor α to A_Y and then add the product $(\alpha A_Y)B_Y^*$ to $N_{X,b}$. Otherwise we apply the scaling factor to B_Y and add the product $A_Y(\bar{\alpha}B_Y)^*$ to $N_{X,b}$. This takes not more than $2k\,|\hat{t}|\,|\hat{s}| + k\min\{|\hat{t}|,|\hat{s}|\}$ operations. Since $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is admissible, either t or s has to be a leaf. If t is a leaf, we have

$$2k\,|\hat{t}|\,|\hat{s}| + k\min\{|\hat{t}|,|\hat{s}|\} \leq r_{\mathcal{I}} 2k\,|\hat{s}| + k\,|\hat{t}| \leq C_{\mathrm{up}} k\,\hat{k}(|\hat{t}| + |\hat{s}|).$$

If s is a leaf, we have

$$2k\,|\hat{t}|\,|\hat{s}| + k\min\{|\hat{t}|,|\hat{s}|\} \leq r_{\mathcal{J}} 2k\,|\hat{t}| + k\,|\hat{s}| \leq C_{\mathrm{up}} k\,\hat{k}(|\hat{t}| + |\hat{s}|).$$

If b is an admissible leaf, we use the function add_rkmatrix. Due to Corollary 5.10, this requires not more than

$$C_{\mathrm{add}} k^2(|\hat{t}| + |\hat{s}|) \leq C_{\mathrm{up}} k\,\hat{k}(|\hat{t}| + |\hat{s}|) \text{ operations.}$$

Now let $n \in \mathbb{N}$ be given such that the estimate holds for all blocks $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with $|\mathcal{T}_b| \leq n$.

Let $b \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with $|\mathcal{T}_b| = n+1$. The block b cannot be a leaf, so the function will call itself for all children b' of b. Due to $|\mathcal{T}_{b'}| \leq n$ for all $b' \in \mathrm{chil}(b)$, we can apply the induction assumption and obtain

$$\begin{aligned} W_{\mathrm{up}}(t,s) &= \sum_{(t',s')\in\mathrm{chil}(t,s)} W_{\mathrm{up}}(t',s') \leq C_{\mathrm{up}} k\,\hat{k} \sum_{b'\in\mathrm{chil}(b)} \sum_{b''=(t'',s'')\in\mathcal{T}_{b'}} |\hat{t}''| + |\hat{s}''| \\ &\leq C_{\mathrm{up}} k\,\hat{k} \sum_{b''=(t'',s'')\in\mathcal{T}_b} |\hat{t}''| + |\hat{s}''|. \end{aligned}$$

Applying Lemma 3.34 yields the final estimate. ■

5.5 Merging

The low-rank update requires us to split a low-rank matrix into submatrices. There are also algorithms that require us to merge several low-rank submatrices into a larger low-rank matrix, e.g., if intermediate results have been computed and have to be combined to form the final result. Let $X_1 \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}_1}, \ldots X_\sigma \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}_\sigma}$ be low-rank matrices with $\sigma \geq 1$ and disjoint index sets $\mathcal{J}_1, \ldots, \mathcal{J}_\sigma$. We are looking for a low-rank approximation of the matrix

$$Z := \begin{pmatrix} X_1 & \ldots & X_\sigma \end{pmatrix} \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}, \qquad \mathcal{J} := \mathcal{J}_1 \cup \ldots \cup \mathcal{J}_\sigma.$$

We assume that the X_ι are given in $\mathcal{R}(k)$-matrix representation, i.e., that there are $A_\iota \in \mathbb{K}^{\mathcal{I}\times k}$, $B_\iota \in \mathbb{K}^{\mathcal{J}_\iota\times k}$ such that

$$X_\iota = A_\iota B_\iota^* \qquad \text{for all } \iota \in [1:\sigma].$$

As in the truncation algorithm, we can make use of QR factorizations to reduce the computational work: we compute isometric matrices $Q_\iota \in \mathbb{K}^{\mathcal{J}_\iota\times k}$ and triangular matrices $R_\iota \in \mathbb{K}^{k\times k}$ such that

$$B_\iota = Q_\iota R_\iota \qquad \text{for all } \iota \in [1:\sigma]$$

and rewrite Z in the form

$$\begin{aligned} Z &= \begin{pmatrix} X_1 & \ldots & X_\sigma \end{pmatrix} = \begin{pmatrix} A_1 B_1^* & \ldots & A_\sigma B_\sigma^* \end{pmatrix} \\ &= \begin{pmatrix} A_1(Q_1R_1)^* & \ldots & A_\sigma(Q_\sigma R_\sigma)^* \end{pmatrix} = \begin{pmatrix} A_1R_1^*Q_1^* & \ldots & A_\sigma R_\sigma^* Q_\sigma^* \end{pmatrix} \\ &= \begin{pmatrix} A_1R_1^* & \ldots & A_\sigma R_\sigma^* \end{pmatrix} \begin{pmatrix} Q_1^* & & \\ & \ddots & \\ & & Q_\sigma^* \end{pmatrix}. \end{aligned}$$

We define

$$\widehat{Z} := \begin{pmatrix} A_1R_1^* & \ldots & A_\sigma R_\sigma^* \end{pmatrix} \in \mathbb{K}^{\mathcal{I}\times(\sigma k)}, \qquad Q := \begin{pmatrix} Q_1 & & \\ & \ddots & \\ & & Q_\sigma \end{pmatrix} \in \mathbb{K}^{\mathcal{J}\times(\sigma k)}$$

and write the equation in the short form

$$Z = \widehat{Z}Q^*.$$

Now we can proceed as before: we find a singular value decomposition $\widehat{Z} = U\Sigma\widehat{V}^*$ of $\widehat{Z}$ and see that

$$Z = \widehat{Z}Q^* = U\Sigma\widehat{V}^*Q^* = U\Sigma(Q\widehat{V})^* = U\Sigma V^*$$

holds with the isometric matrix $V := Q\widehat{V}$.

```
procedure rowmerge_rkmatrix ($X_1,\dots,X_\sigma$, var $Y$);
  { $\mathcal{R}(k)$-representations $X_\iota = A_\iota B_\iota^*$ for all $\iota \in [1:\sigma]$ }
  { $\mathcal{R}(k)$-representation $Y = A_Y B_Y^*$ for the result }
  for $\iota \in [1:\sigma]$ do
    Compute QR factorization $B_\iota = Q_\iota R_\iota$ with $Q_\iota \in \mathbb{K}^{\mathcal{J}_\iota \times k}$, $R_\iota \in \mathbb{K}^{k\times k}$;
  $\widehat{Z} \leftarrow \begin{pmatrix} A_1 R_1^* & A_2 R_2^* & \dots & A_\sigma R_\sigma^* \end{pmatrix} \in \mathbb{K}^{\mathcal{I}\times(\sigma k)}$
  Compute singular value decomposition $\widehat{Z} = U\Sigma\widehat{V}^*$, rank $p \le \sigma k$;
  Choose rank $\tilde{k}$;
  for $\iota \in [1:\sigma]$ do
    $V|_{\mathcal{J}_\iota \times \tilde{k}} \leftarrow Q_\iota (\widehat{V}_\iota)|_{k\times\tilde{k}}$;
  $A_Y \leftarrow U|_{\mathcal{I}\times\tilde{k}} \Sigma|_{\tilde{k}\times\tilde{k}}$;    $B_Y \leftarrow V|_{\mathcal{J}\times\tilde{k}}$
end
```

Figure 5.6: Merging $\mathcal{R}(k)$-matrices in a row, Y is overwritten by a rank-$\tilde{k}$ approximation of Z.

Since Q is a block-diagonal matrix, we can apply it efficiently to $\widehat{V} \in \mathbb{K}^{(\sigma k)\times p}$ by splitting

$$\widehat{V} = \begin{pmatrix} \widehat{V}_1 \\ \vdots \\ \widehat{V}_\sigma \end{pmatrix} \qquad \text{with} \qquad \widehat{V}_\iota \in \mathbb{K}^{k\times p} \qquad \text{for all } \iota \in [1:\sigma]$$

and using

$$V = \begin{pmatrix} Q_1 \widehat{V}_1 \\ \vdots \\ Q_\sigma \widehat{V}_\sigma \end{pmatrix} \in \mathbb{K}^{\mathcal{J}\times p}.$$

Obtaining an optimal low-rank approximation is now straightforward, the resulting algorithm is summarized in Figure 5.6.

We also need an algorithm for merging column matrices. Let $X_1 \in \mathbb{K}^{\mathcal{I}_1\times\mathcal{J}}, X_2 \in \mathbb{K}^{\mathcal{I}_2\times\mathcal{J}}, \dots, X_\sigma \in \mathbb{K}^{\mathcal{I}_\sigma\times\mathcal{J}}$ be low-rank matrices with $\sigma \ge 1$ and disjoint index sets $\mathcal{I}_1,\dots,\mathcal{I}_\sigma$. We are looking for a low-rank approximation of

$$Z := \begin{pmatrix} X_1 \\ \vdots \\ X_\sigma \end{pmatrix} \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}, \qquad\qquad \mathcal{I} := \mathcal{I}_1 \cup \mathcal{I}_2 \cup \dots \cup \mathcal{I}_\sigma.$$

Applying the same procedure as before to Z^* instead of Z yields the algorithm summarized in Figure 5.7: we use QR factorizations $Q_\iota R_\iota = A_\iota$, construct $\widehat{Z} \in \mathbb{K}^{\mathcal{J}\times(\sigma k)}$, and obtain a singular value decomposition of Z^* instead of Z.

Essentially we only have to switch the roles of the low-rank factors A and B in both the input matrices and the result in order to merge rows instead of columns.

```
procedure colmerge_rkmatrix ($X_1$,...,$X_\sigma$, var $Y$);
  { $\mathcal{R}(k)$-representations $X_\iota = A_\iota B_\iota^*$ for all $\iota \in [1:\sigma]$ }
  { $\mathcal{R}(k)$-representation $Y = A_Y B_Y^*$ for the result }
  for $\iota \in [1:\sigma]$ do
    Compute QR factorization $A_\iota = Q_\iota R_\iota$ with $Q_\iota \in \mathbb{K}^{\mathcal{I}_\iota \times k}$, $R_\iota \in \mathbb{K}^{k \times k}$;
  $\widehat{Z} \leftarrow \begin{pmatrix} B_1 R_1^* & \cdots & B_\sigma R_\sigma^* \end{pmatrix} \in \mathbb{K}^{\mathcal{J} \times (\sigma k)}$
  Compute singular value decomposition $\widehat{Z} = U \Sigma \widehat{V}^*$, rank $p \le \sigma k$;
  Choose rank $\tilde{k}$;
  for $\iota \in [1:\sigma]$ do
    $V|_{\mathcal{I}_\iota \times \tilde{k}} \leftarrow Q_\iota (\widehat{V}_\iota)|_{k \times \tilde{k}}$;
  $B \leftarrow U|_{\mathcal{J} \times \tilde{k}}$;    $A \leftarrow V|_{\mathcal{I} \times \tilde{k}} \Sigma|_{\tilde{k} \times \tilde{k}}$
end
```

Figure 5.7: Merging $\mathcal{R}(k)$-matrices in a column, Y is overwritten by a rank-$\tilde{k}$ approximation of Z.

Lemma 5.12 (Complexity of merging matrices) *Let $\sigma \in \mathbb{N}$, and let $X_1 \in \mathbb{K}^{\mathcal{I} \times \mathcal{J}_1}$, ..., $X_\sigma \in \mathbb{K}^{\mathcal{I} \times \mathcal{J}_\sigma}$ be rank-k matrices in $\mathcal{R}(k)$-matrix representation, let $\tilde{k} \in [0:\sigma k]$. The algorithm `rowmerge_rkmatrix` computes an $\mathcal{R}(\tilde{k})$-matrix approximation of*

$$Z = \begin{pmatrix} X_1 & \dots & X_\sigma \end{pmatrix} \in \mathbb{K}^{\mathcal{I} \times \mathcal{J}}, \qquad \mathcal{J} := \mathcal{J}_1 \cup \ldots \cup \mathcal{J}_\sigma,$$

in not more than

$$C_{mg} \sigma^2 k^2 (|\mathcal{I}| + |\mathcal{J}|) \text{ operations}$$

with $C_{mg} := \max\{2C_{qr}, 3 + C_{svd}\}$.

Let $X_1 \in \mathbb{K}^{\mathcal{I}_1 \times \mathcal{J}}$, $X_2 \in \mathbb{K}^{\mathcal{I}_2 \times \mathcal{J}}$, ..., $X_\sigma \in \mathbb{K}^{\mathcal{I}_\sigma \times \mathcal{J}}$ be rank-k-matrices in $\mathcal{R}(k)$-matrix representation, let $\tilde{k} \in [0:\sigma k]$. The algorithm `colmerge_rkmatrix` computes an $\mathcal{R}(\tilde{k})$-matrix approximation of

$$Z = \begin{pmatrix} X_1 \\ \vdots \\ X_\sigma \end{pmatrix} \in \mathbb{K}^{\mathcal{I} \times \mathcal{J}}, \qquad \mathcal{I} := \mathcal{I}_1 \cup \ldots \cup \mathcal{I}_\sigma$$

in not more than

$$C_{mg} \sigma^2 k^2 (|\mathcal{I}| + |\mathcal{J}|) \text{ operations.}$$

Proof. Due to Assumption 5.7, the QR factorizations $B_\iota = Q_\iota R_\iota$ can be constructed in not more than

$$\sum_{\iota=1}^{\sigma} C_{\mathrm{qr}} |\mathcal{J}_\iota| \, k^2 = C_{\mathrm{qr}} k^2 \, |\mathcal{J}| \text{ operations.}$$

The matrix

$$\widehat{Z} = \begin{pmatrix} A_1 R_1^* & A_2 R_2^* & \dots & A_\sigma R_\sigma^* \end{pmatrix}$$

can be computed by multiplying $\mathcal{I} \times k$ and $k \times k$ matrices in

$$\sum_{\iota=1}^{\sigma} 2k^2 \, |\mathcal{I}| = 2\sigma k^2 \, |\mathcal{I}| \leq 2\sigma^2 k^2 \, |\mathcal{I}| \text{ operations,}$$

and its singular value decomposition can be found in not more than

$$C_{\text{svd}} |\mathcal{I}| \, (\sigma k)^2 = C_{\text{svd}} \sigma^2 k^2 \, |\mathcal{I}| \text{ operations.}$$

Applying the matrices Q_ι to $\widehat{V}_\iota|_{k \times \tilde{k}}$ requires not more than

$$\sum_{\iota=1}^{\sigma} C_{\text{qr}} |\mathcal{J}_\iota| \, k \, \tilde{k} = C_{\text{qr}} |\mathcal{J}| \, k \, \tilde{k} \leq C_{\text{qr}} \sigma k^2 \, |\mathcal{J}| \text{ operations}$$

due to $\tilde{k} \leq \sigma k$, and the product of $U|_{\mathcal{I} \times \tilde{k}}$ and $\Sigma|_{\tilde{k} \times \tilde{k}}$ can be computed in

$$|\mathcal{I}| \, \tilde{k} \leq |\mathcal{I}| \, (\sigma k) \leq \sigma^2 k^2 \, |\mathcal{I}| \text{ operations.}$$

Adding these estimates yields the required bound.

The same arguments can be applied to `colmerge_rkmatrix`. ■

Remark 5.13 (Inductive merge) *Typical cluster strategies lead to clusters with two or three children, i.e., we will have $\sigma \in \{2, 3\}$, and the quadratic dependence of the computational work on σ does not matter too much.*

If we want to merge a larger number of submatrices, we may consider merging them step by step*: we first compute the singular value decomposition*

$$\begin{pmatrix} A_1 R_1^* & A_2 R_2^* \end{pmatrix} = U_{12} \Sigma_{12} \widehat{V}_{12}^*.$$

Now we truncate to $\widetilde{\Sigma}_{12} \in \mathbb{K}^{\tilde{k} \times \tilde{k}}$ and $\widetilde{V}_{12} \in \mathbb{K}^{(2k) \times \tilde{k}}$.

In the next step, we have

$$\begin{pmatrix} A_1 R_1^* & A_2 R_2^* & A_3 R_3^* \end{pmatrix} \approx \begin{pmatrix} U_{1,2} \widetilde{\Sigma}_{1,2} \widetilde{V}_{1,2}^* & A_3 R_3^* \end{pmatrix} = \begin{pmatrix} U_{1,2} \widetilde{\Sigma}_{1,2} & A_3 R_3^* \end{pmatrix} \begin{pmatrix} \widetilde{V}_{1,2} & \\ & I \end{pmatrix}^*$$

We have $U_{1,2} \widetilde{\Sigma}_{1,2} \in \mathbb{K}^{\mathcal{I} \times \tilde{k}}$ and $A_3 R_3^* \in \mathbb{K}^{\mathcal{I} \times k}$, so the singular value decomposition

$$\begin{pmatrix} U_{1,2} \widetilde{\Sigma}_{1,2} & A_3 R_3^* \end{pmatrix} = U_{1,3} \Sigma_{1,3} \widehat{V}_{1,3}^*$$

can be computed in not more than $C_{svd}(k + \tilde{k})^2 \, |\mathcal{I}|$ operations. Again we can truncate to $\widetilde{\Sigma}_{1,3} \in \mathbb{K}^{\tilde{k} \times \tilde{k}}$ and $\widetilde{V}_{1,3} \in \mathbb{K}^{(k+\tilde{k}) \times \tilde{k}}$ and find

$$\begin{pmatrix} A_1 R_1^* & A_2 R_2^* & A_3 R_3^* \end{pmatrix} \approx U_{1,3} \widetilde{\Sigma}_{1,3} \widetilde{V}_{1,3}^* \begin{pmatrix} \widetilde{V}_{1,2} & \\ & I \end{pmatrix}^* .$$

If we keep repeating the procedure, we only require $\mathcal{O}(\sigma(k + \tilde{k})^2(|\mathcal{I}| + |\mathcal{J}|))$ operations, the computational work only grows linearly with σ, but the intermediate truncation steps influence the accuracy and we will not always compute the best low-rank approximation.

5.6 Matrix multiplication

With efficient algorithms for matrix-vector multiplications, low-rank updates, and merging submatrices at our disposal, we can face the challenge of finding an efficient algorithm for approximating the product of two $\mathcal{H}$-matrices.

We assume that $X \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$, $Y \in \mathbb{K}^{\mathcal{J}\times\mathcal{K}}$, and $Z \in \mathbb{K}^{\mathcal{I}\times\mathcal{K}}$ are $\mathcal{H}$-matrices of local rank k for the admissible block trees $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, $\mathcal{T}_{\mathcal{J}\times\mathcal{K}}$ and $\mathcal{T}_{\mathcal{I}\times\mathcal{K}}$, respectively. Let (A_X, B_X, N_X), (A_Y, B_Y, N_Y) and (A_Z, B_Z, N_Z) be $\mathcal{H}$-matrix representations of X, Y, and Z.

We let $\alpha \in \mathbb{K}$ and consider the update operation

$$Z \leftarrow Z + \alpha XY.$$

As in the case of the matrix-vector multiplication, we split the matrices into suitable submatrices and develop an algorithm for performing the update

$$Z|_{\hat{t}\times\hat{r}} \leftarrow Z|_{\hat{t}\times\hat{r}} + \alpha X|_{\hat{t}\times\hat{s}} Y|_{\hat{s}\times\hat{r}} \tag{5.6}$$

for suitable clusters $t \in \mathcal{T}_{\mathcal{I}}$, $s \in \mathcal{T}_{\mathcal{J}}$ and $r \in \mathcal{T}_{\mathcal{K}}$ such that $(t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ and $(s,r) \in \mathcal{T}_{\mathcal{J}\times\mathcal{K}}$.

Case 1: (s,r) is a leaf. If $b = (s,r)$ is a leaf, the rank of $Y|_{\hat{s}\times\hat{r}}$ is bounded, either because b is admissible, i.e., $Y|_{\hat{s}\times\hat{r}} = A_{Y,b}B_{Y,b}^*$, or because either s or r is a leaf, cf. Definition 3.28, and therefore $|\hat{s}|$ or $|\hat{r}|$ are small.

Case 1a: (s,r) is an admissible leaf. If $b = (s,r)$ is an admissible leaf, we have

$$Y|_{\hat{s}\times\hat{r}} = A_{Y,b}B_{Y,b}^*$$

and therefore

$$X|_{\hat{t}\times\hat{s}} Y|_{\hat{s}\times\hat{r}} = X|_{\hat{t}\times\hat{s}} A_{Y,b}B_{Y,b}^*.$$

We can compute

$$\widehat{A} := X|_{\hat{t}\times\hat{s}} A_{Y,b}$$

by applying the algorithm `addeval_hmatrix` (cf. Figure 5.1) to the matrix $A_{Y,b}$. The update (5.6) takes the form

$$Z|_{\hat{t}\times\hat{r}} \leftarrow Z|_{\hat{t}\times\hat{r}} + \alpha \widehat{A} B_{Y,b}^*$$

of a low-rank update that can be handled by the algorithm `add_rkmatrix_hmatrix` (cf. Figure 5.5).

Case 1b: (s,r) is an inadmissible leaf. If $b = (s,r)$ is an inadmissible leaf, either s or r has to be a leaf cluster due to Definition 3.28.

If s is a leaf, we have $|\hat{s}| \leq r_{\mathcal{J}}$ and the product

$$X|_{\hat{t}\times\hat{s}} Y|_{\hat{s}\times\hat{r}} = X|_{\hat{t}\times\hat{s}} N_{Y,b}$$

is already a factorized low-rank representation. We only have to convert the $\mathcal{H}$-matrix into a standard matrix, e.g., by applying `addeval_hmatrix` to the identity matrix $I_{\hat{s}} \in \mathbb{K}^{\hat{s}\times\hat{s}}$ to get

$$\widehat{A} \leftarrow X|_{\hat{t}\times\hat{s}} I_{\hat{s}} \in \mathbb{K}^{\hat{t}\times\hat{s}}.$$

Once we have $\widehat{A}$ and $N_{Y,b}$ at our disposal, the $\mathcal{R}(\hat{s})$-matrix

$$X|_{\hat{t}\times\hat{s}} Y|_{\hat{s}\times\hat{r}} = \widehat{A} N_{Y,b} = \widehat{A}(N_{Y,b}^*)^*$$

with $|\hat{s}| \leq r_{\mathcal{J}}$ can be added to $Z|_{\hat{t}\times\hat{r}}$ using the function `add_rkmatrix_hmatrix`.

If r is a leaf, we have $|\hat{r}| \leq r_{\mathcal{K}}$ and compute

$$\widehat{A} := X|_{\hat{t}\times\hat{s}} Y|_{\hat{s}\times\hat{r}} = \alpha X|_{\hat{t}\times\hat{s}} N_{Y,b}$$

by applying `addeval_hmatrix` to the matrix $N_{Y,b}$. Now

$$X|_{\hat{t}\times\hat{s}} Y|_{\hat{s}\times\hat{r}} = \widehat{A} I_{\hat{r}}^*$$

is a factorized low-rank representation of rank not larger than $|\hat{r}| \leq r_{\mathcal{K}}$. and we can perform the required update again by using `add_rkmatrix_hmatrix`.

Case 2: (t,s) is a leaf. If $b = (t,s)$ is a leaf, the rank of $X|_{\hat{t}\times\hat{s}}$ is bounded, again either because b is admissible or because the number of rows or columns is small.

Case 2a: (t,s) is an admissible leaf. If $b = (t,s)$ is an admissible leaf, we have

$$X|_{\hat{t}\times\hat{s}} = A_{X,b} B_{X,b}^*$$

and therefore

$$X|_{\hat{t}\times\hat{s}} Y|_{\hat{s}\times\hat{r}} = A_{X,b} B_{X,b}^* Y|_{\hat{s}\times\hat{r}} = A_{X,b}(Y|_{\hat{s}\times\hat{r}}^* B_{X,b})^*.$$

We can compute

$$\widehat{B} := Y|_{\hat{s}\times\hat{r}}^* B_{X,b}$$

by applying the algorithm `addevaltrans_hmatrix` (cf. Figure 5.2) to the matrix $B_{X,b}$. The update (5.6) takes the form

$$Z|_{\hat{t}\times\hat{r}} \leftarrow Z|_{\hat{t}\times\hat{r}} + \alpha A_{X,b} \widehat{B}^*$$

of a low-rank update that can be handled by the algorithm `add_rkmatrix_hmatrix` (cf. Figure 5.5).

Case 2b: (t,s) is an inadmissible leaf. If $b = (t,s)$ is an inadmissible leaf, either t or s has to be a leaf cluster due to Definition 3.28.

If s is a leaf, we have $|\hat{s}| \leq r_{\mathcal{J}}$ and the product is already a factorized low-rank representation. We convert the $\mathcal{H}$-matrix $Y|_{\hat{s}\times\hat{r}}$ into a standard matrix, e.g., by applying `addevaltrans_hmatrix` to the identity matrix $I_{\hat{s}} \in \mathbb{K}^{\hat{s}\times\hat{s}}$, and obtain

$$\widehat{B} := Y|_{\hat{s}\times\hat{r}}^* I_{\hat{s}}$$

such that

$$X|_{\hat{t}\times\hat{s}} Y|_{\hat{s}\times\hat{r}} = N_{X,b}\widehat{B}^*.$$

Once again we can use the algorithm `add_rkmatrix_hmatrix` (cf. Figure 5.5) to add this low-rank matrix to $Z|_{\hat{t}\times\hat{r}}$.

If t is a leaf, we have $|\hat{t}| \leq r_{\mathcal{I}}$. In this case, we compute

$$\widehat{B} := (X|_{\hat{t}\times\hat{s}} Y|_{\hat{s}\times\hat{r}})^* = Y|_{\hat{s}\times\hat{r}}^* N_{X,b}^*$$

by applying `addevaltrans_hmatrix` (cf. Figure 5.2) to the matrix $N_{X,b}^*$. Now

$$X|_{\hat{t}\times\hat{s}} Y|_{\hat{s}\times\hat{r}} = I_{\hat{t}}\widehat{B}^*$$

is a factorized low-rank representation of the product, with a rank bounded by $|\hat{t}| \leq r_{\mathcal{I}}$, and we can perform the required update again by using `add_rkmatrix_hmatrix`.

Case 3: (t,s) and (s,r) are not leaves. We can handle this case be recursion: with the notations of Lemma 3.21, we have

$$\operatorname{chil}(t,s) = \operatorname{chil}^+(t) \times \operatorname{chil}^+(s), \qquad \operatorname{chil}(s,r) = \operatorname{chil}^+(s) \times \operatorname{chil}^+(r),$$

and can replace (5.6) by the updates

$$Z|_{\hat{t}'\times\hat{r}'} \leftarrow Z|_{\hat{t}'\times\hat{r}'} + \alpha X|_{\hat{t}'\times\hat{s}'} Y|_{\hat{s}'\times\hat{r}'} \quad \text{for all } t' \in \operatorname{chil}^+(t),\ s' \in \operatorname{chil}^+(s),\ r' \in \operatorname{chil}^+(r).$$

If (t,r) is not a leaf, we have

$$\operatorname{chil}(t,r) = \operatorname{chil}^+(t) \times \operatorname{chil}^+(r)$$

and can apply recursion directly.

Otherwise, i.e., if (t,r) is a leaf, we have to split $Z|_{\hat{t}\times\hat{r}}$ into auxiliary submatrices $Z|_{\hat{t}'\times\hat{r}'}$ for all $t' \in \operatorname{chil}^+(t)$ and $r' \in \operatorname{chil}^+(s)$, treat these submatrices by recursion, and the merge them again.

If (t,r) is an inadmissible leaf, splitting means copying submatrices, and merging means combining them again into the final result. An alternative is to work directly with submatrices of $N_{Z,(t,r)}$ and avoid copy operations.

If (t,r) is an admissible leaf, we have $Z|_{\hat{t}\times\hat{r}} = A_{Z,(t,r)} B_{Z,(t,r)}^*$ and find

$$Z|_{\hat{t}'\times\hat{r}'} = A_{Z,(t,r)}|_{\hat{t}'\times k} B_{Z,(t,r)}|_{\hat{r}'\times k}^* \qquad \text{for all } t' \in \operatorname{chil}^+(t),\ s' \in \operatorname{chil}^+(s),$$

we can use `rowmerge_rkmatrix` (cf. Figure 5.6) and `colmerge_rkmatrix` (cf. Figure 5.7) to merge the submatrices.

```
procedure addmul_hmatrix($t$, $s$, $r$, $\alpha$, $X$, $Y$, var $Z$);
  { $\mathcal{H}$-matrix representations $(A_X, B_X, N_X)$, $(A_Y, B_Y, N_Y)$,
    and $(A_Z, B_Z, N_Z)$ of $X$, $Y$, and $Z$ }
  if $(s,r) \in \mathcal{L}^+_{\mathcal{J}\times\mathcal{K}}$ then begin
    $\widehat{A} \leftarrow 0 \in \mathbb{K}^{\hat{t}\times k}$; addeval_hmatrix(1, $X$, $(t,s)$, $A_{Y,(s,r)}$, $\widehat{A}$);
    add_rkmatrix_hmatrix($(t,r)$, $\alpha$, $\widehat{A}$, $B_{Y,(s,r)}$, $Z$)
  end else if $(s,r) \in \mathcal{L}^-_{\mathcal{J}\times\mathcal{K}}$ then begin
    if $|\hat{s}| \le |\hat{r}|$ then begin
      $\widehat{A} \leftarrow 0 \in \mathbb{K}^{\hat{t}\times\hat{s}}$; addeval_hmatrix(1, $X$, $(t,s)$, $I_{\hat{s}}$, $\widehat{A}$); $\widehat{B} \leftarrow N^*_{Y,(s,r)}$
    end else begin
      $\widehat{A} \leftarrow 0 \in \mathbb{K}^{\hat{t}\times\hat{r}}$; addeval_hmatrix(1, $X$, $(t,s)$, $N_{Y,(s,r)}$, $\widehat{A}$); $\widehat{B} \leftarrow I_{\hat{r}}$
    end;
    add_rkmatrix_hmatrix($(t,r)$, $\alpha$, $\widehat{A}$, $\widehat{B}$, $Z$)
  end else if $(t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ then begin
    $\widehat{B} \leftarrow 0 \in \mathbb{K}^{\hat{s}\times k}$; addevaltrans_hmatrix(1, $Y$, $(s,r)$, $B_{X,(t,s)}$, $\widehat{B}$);
    add_rkmatrix_hmatrix($(t,r)$, $\alpha$, $A_{X,(t,s)}$, $\widehat{B}$, $Z$)
  end else if $(t,s) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$ then
    if $|\hat{s}| \le |\hat{t}|$ then begin
      $\widehat{B} \leftarrow 0 \in \mathbb{K}^{\hat{r}\times\hat{s}}$; addevaltrans_hmatrix(1, $Y$, $(s,r)$, $I_{\hat{s}}$, $\widehat{B}$); $\widehat{A} \leftarrow N_{X,(t,s)}$
    end else begin
      $\widehat{B} \leftarrow 0 \in \mathbb{K}^{\hat{r}\times\hat{t}}$; addevaltrans_hmatrix(1, $Y$, $(s,r)$, $N^*_{X,(t,s)}$, $\widehat{B}$); $\widehat{A} \leftarrow I_{\hat{t}}$
    end;
    add_rkmatrix_hmatrix($(t,r)$, $\alpha$, $\widehat{A}$, $\widehat{B}$, $Z$)
  end else
    if $(t,r) \notin \mathcal{T}_{\mathcal{I}\times\mathcal{K}} \setminus \mathcal{L}_{\mathcal{I}\times\mathcal{K}}$ then begin
      Split $Z|_{\hat{t}\times\hat{r}}$ into submatrices;
      for $t' \in \operatorname{chil}^+(t)$, $s' \in \operatorname{chil}^+(s)$, $r' \in \operatorname{chil}^+(r)$ do
        addmul_hmatrix($t'$, $s'$, $r'$, $\alpha$, $X$, $Y$, $Z$);
      Merge submatrices into $Z|_{\hat{t}\times\hat{r}}$
    end else
      for $t' \in \operatorname{chil}^+(t)$, $s' \in \operatorname{chil}^+(s)$, $r' \in \operatorname{chil}^+(r)$ do
        addmul_hmatrix($t'$, $s'$, $r'$, $\alpha$, $X$, $Y$, $Z$)
end
```

Figure 5.8: $\mathcal{H}$-matrix multiplication, $Z|_{\hat{t}\times\hat{r}}$ is overwritten by an $\mathcal{H}$-matrix approximation of $Z|_{\hat{t}\times\hat{r}} + \alpha X|_{\hat{t}\times\hat{s}} Y|_{\hat{s}\times\hat{r}}$.

Remark 5.14 (Conversion) *In a practical implementation, it is advisable to treat inadmissible leaves by separate algorithms: converting an $\mathcal{H}$-matrix directly to a standard matrix is simpler than using `addeval_hmatrix`, and approximating a standard matrix using the singular value decomposition is simpler than using `add_rkmatrix_hmatrix`.*

5.7 Complexity of the matrix multiplication

In order to analyze the complexity of the multiplication algorithm given in Figure 5.8, we require a number of fairly straightforward assumptions and definitions.

- Assume that the block trees $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, $\mathcal{T}_{\mathcal{J}\times\mathcal{K}}$, and $\mathcal{T}_{\mathcal{I}\times\mathcal{K}}$ are admissible and C_{sp}-sparse.
- Let $p_{\mathcal{I}}$, $p_{\mathcal{J}}$, and $p_{\mathcal{K}}$ denote the depths of the cluster trees $\mathcal{T}_{\mathcal{I}}$, $\mathcal{T}_{\mathcal{J}}$, and $\mathcal{T}_{\mathcal{K}}$, and define the maximal depth by
$$\hat{p} := \max\{p_{\mathcal{I}}, p_{\mathcal{J}}, p_{\mathcal{K}}\}. \tag{5.7}$$
Lemma 3.33 states that $\hat{p}$ is also an upper bound for the depths of the block trees $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, $\mathcal{T}_{\mathcal{J}\times\mathcal{K}}$, and $\mathcal{T}_{\mathcal{I}\times\mathcal{K}}$.
- Let $r_{\mathcal{I}}$, $r_{\mathcal{J}}$ and $r_{\mathcal{K}}$ denote the resolutions of the cluster trees $\mathcal{T}_{\mathcal{I}}$, $\mathcal{T}_{\mathcal{J}}$, $\mathcal{T}_{\mathcal{K}}$. Then the rank of both admissible and inadmissible leaf blocks of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, $\mathcal{T}_{\mathcal{J}\times\mathcal{K}}$, and $\mathcal{T}_{\mathcal{I}\times\mathcal{K}}$ is bounded by
$$\hat{k} := \max\{k, r_{\mathcal{I}}, r_{\mathcal{J}}, r_{\mathcal{K}}\}. \tag{5.8}$$
- Let σ denote an upper bound for the number of children of clusters in $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{K}}$, i.e., assume
$$|\operatorname{chil}(t)| \le \sigma, \qquad |\operatorname{chil}(r)| \le \sigma \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}},\ r \in \mathcal{T}_{\mathcal{K}}. \tag{5.9}$$

In order to find a bound for the computational work involved in the approximate $\mathcal{H}$-matrix multiplication function `addmul_hmatrix`, we consider the triples (t, s, r) that appear as its parameters. Following the recursive calls yields a new tree structure: our algorithm stops the recursion only if either (t, s) or (s, r) is a leaf.

Definition 5.15 (Product tree) *A tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}} = (V, \varrho, E)$ is called a* product tree *for $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{\mathcal{J}\times\mathcal{K}}$ if*

- *the nodes are triples of clusters, i.e.,*
$$V \subseteq \mathcal{T}_{\mathcal{I}} \times \mathcal{T}_{\mathcal{J}} \times \mathcal{T}_{\mathcal{K}}, \tag{5.10a}$$
- *the root consists of the roots of $\mathcal{T}_{\mathcal{I}}$, $\mathcal{T}_{\mathcal{J}}$ and $\mathcal{T}_{\mathcal{K}}$, i.e.,*
$$\operatorname{root}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}) = (\operatorname{root}(\mathcal{T}_{\mathcal{I}}), \operatorname{root}(\mathcal{T}_{\mathcal{J}}), \operatorname{root}(\mathcal{T}_{\mathcal{K}})), \ \textit{and} \tag{5.10b}$$
- *the children of $p = (t, s, r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ are given by*
$$\operatorname{chil}(p) = \begin{cases} \operatorname{chil}^+(t) \times \operatorname{chil}^+(s) \times \operatorname{chil}^+(r) & \textit{if } (t,s) \notin \mathcal{L}_{\mathcal{I}\times\mathcal{J}} \wedge (s,r) \notin \mathcal{L}_{\mathcal{J}\times\mathcal{K}}, \\ \emptyset & \textit{otherwise.} \end{cases} \tag{5.10c}$$

Lemma 5.16 (Relation to block trees) *We have* $(t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ *if and only if* $(t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$*,* $(s,r) \in \mathcal{T}_{\mathcal{J}\times\mathcal{K}}$ *and* $\operatorname{level}(t,s) = \operatorname{level}(s,r)$ *hold.*

Proof. We first prove

$$(t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}} \Longrightarrow ((t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}} \wedge (s,r) \in \mathcal{T}_{\mathcal{J}\times\mathcal{K}} \wedge \operatorname{level}(t,s) = \operatorname{level}(s,r)) \tag{5.11}$$

by induction on $\operatorname{level}(t,s,r)$.

If $\operatorname{level}(t,s,r) = 0$, we have $t = \operatorname{root}(\mathcal{T}_\mathcal{I})$, $s = \operatorname{root}(\mathcal{T}_\mathcal{J})$ and $r = \operatorname{root}(\mathcal{T}_\mathcal{K})$, and (3.13a) implies $(t,s) = \operatorname{root}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}})$, $(s,r) = \operatorname{root}(\mathcal{T}_{\mathcal{J}\times\mathcal{K}})$, and $\operatorname{level}(t,s) = 0 = \operatorname{level}(s,r)$.

Let now $n \in \mathbb{N}_0$ be given such that (5.11) holds for all $(t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ with $\operatorname{level}(t,s,r) = n$.

Let $(t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ with $\operatorname{level}(t,s,r) = n+1$. Then there has to be a parent $(t^+,s^+,r^+) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ with $\operatorname{level}(t^+,s^+,r^+) = n$. By the induction assumption, we have $(t^+,s^+) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, $(s^+,r^+) \in \mathcal{T}_{\mathcal{J}\times\mathcal{K}}$, and $\operatorname{level}(t^+,s^+) = \operatorname{level}(s^+,r^+)$.

Due to (5.10c), we have $t \in \operatorname{chil}^+(t^+)$, $s \in \operatorname{chil}^+(s^+)$ and $r \in \operatorname{chil}^+(r^+)$, and (3.13b) yields $(t,s) \in \operatorname{chil}(t^+,s^+) \subseteq \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ and $(s,r) \in \operatorname{chil}(s^+,r^+) \subseteq \mathcal{T}_{\mathcal{J}\times\mathcal{K}}$. Due to $\operatorname{level}(t^+,s^+) = \operatorname{level}(s^+,r^+)$, this implies also $\operatorname{level}(t,s) = \operatorname{level}(s,r)$.

Now we prove

$$((t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}} \wedge (s,r) \in \mathcal{T}_{\mathcal{J}\times\mathcal{K}} \wedge \operatorname{level}(t,s) = \operatorname{level}(s,r)) \Longrightarrow (t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}} \tag{5.12}$$

by induction on $\operatorname{level}(t,s)$.

If $\operatorname{level}(t,s) = 0$, $\operatorname{level}(t,s) = \operatorname{level}(s,r)$ yields $\operatorname{level}(s,r) = 0$, and (3.13a) implies $t = \operatorname{root}(\mathcal{T}_\mathcal{I})$, $s = \operatorname{root}(\mathcal{T}_\mathcal{J})$ and $r = \operatorname{root}(\mathcal{T}_\mathcal{K})$. Due to (5.10b), we have $(t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$.

Let now $n \in \mathbb{N}_0$ be given such that (5.12) holds for all $(t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with $\operatorname{level}(t,s) = n$.

Let $(t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ and $(s,r) \in \mathcal{T}_{\mathcal{J}\times\mathcal{K}}$ with $\operatorname{level}(t,s) = \operatorname{level}(s,r) = n+1$. Then there have to be parents $(t^+,s^+) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ and $(s^+,r^+) \in \mathcal{T}_{\mathcal{J}\times\mathcal{K}}$ of (t,s) and (s,r) with $\operatorname{level}(t^+,s^+) = n$ and $\operatorname{level}(s^+,r^+) = n$. We can apply the induction assumption to obtain $(t^+,s^+,r^+) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$.

Due to $(t,s) \in \operatorname{chil}(t^+,s^+)$ and $(s,r) \in \operatorname{chil}(s^+,r^+)$, both (t^+,s^+) and (s^+,r^+) cannot be leaves, and (5.10c) yields $\operatorname{chil}(t^+,s^+,r^+) = \operatorname{chil}^+(t^+) \times \operatorname{chil}^+(s^+) \times \operatorname{chil}^+(r^+)$. With (3.13b), we find $t \in \operatorname{chil}^+(t^+)$, $s \in \operatorname{chil}^+(s^+)$ and $r \in \operatorname{chil}^+(r^+)$, and conclude $(t,s,r) \in \operatorname{chil}(t^+,s^+,r^+) \subseteq \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$. ■

The product tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ describes our algorithm's recursive structure. In order to derive a bound for the complexity, we have to investigate how many operations are performed for each of its nodes. In order to keep the notation short, we introduce the abbreviation

$$B_{\mathcal{I}\times\mathcal{J}}(t,s) := \begin{cases} \sum_{(t',s')\in\mathcal{T}_{(t,s)}} |\hat{t}'| + |\hat{s}'| & \text{if } (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}, \\ |\hat{t}| + |\hat{s}| & \text{otherwise} \end{cases} \qquad \text{for all } t \in \mathcal{T}_\mathcal{I},\ s \in \mathcal{T}_\mathcal{J}, \tag{5.13}$$

where the case $(t,s) \notin \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is required to handle the special case of temporarily created auxiliary matrices appearing in Case 3 of the algorithm. Theorem 5.6 and Lemma 5.11

can be written in the short form

$$\begin{aligned} W_{\mathrm{mv}}(t,s,\mathcal{M}) &\le 2\hat{k}\,|\mathcal{M}|\,B_{\mathcal{I}\times\mathcal{J}}(t,s) && \text{for all } (t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}} \text{ and finite sets } \mathcal{M},\\ W_{\mathrm{up}}(t,s) &\le C_{\mathrm{up}}\hat{k}^2 B_{\mathcal{I}\times\mathcal{J}}(t,s) && \text{for all } (t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}. \end{aligned}$$

We define $B_{\mathcal{J}\times\mathcal{K}}$ and $B_{\mathcal{I}\times\mathcal{K}}$ in the same way for the block trees $\mathcal{T}_{\mathcal{J}\times\mathcal{K}}$ and $\mathcal{T}_{\mathcal{I}\times\mathcal{K}}$.

Lemma 5.17 (Leaves) *Let $(t,s,r)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ be a leaf of the product tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$. If the function* ***addmul_hmatrix*** *is called with the parameters t, s and r, it performs not more than*

$$C_{mml}\hat{k}^2\left(B_{\mathcal{I}\times\mathcal{J}}(t,s)+B_{\mathcal{J}\times\mathcal{K}}(s,r)+B_{\mathcal{I}\times\mathcal{K}}(t,r)\right) \text{ operations}$$

with $C_{mml}:=\max\{C_{up},2\}$.

Proof. Since (t,s,r) is a leaf, we have $(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}$ or $(s,r)\in\mathcal{L}_{\mathcal{J}\times\mathcal{K}}$.

Case 1a: If $b:=(s,r)\in\mathcal{L}^{+}_{\mathcal{J}\times\mathcal{K}}$, we compute $\widehat{A}=X|_{\hat{t}\times\hat{s}}A_{Y,b}$ by using the function `addeval_hmatrix` for the matrix $A_{Y,b}$ with k columns. Theorem 5.6 yields that this takes not more than

$$W_{\mathrm{mv}}(t,s,[1:k])\le 2\hat{k}^2B_{\mathcal{I}\times\mathcal{J}}(t,s) \text{ operations.}$$

If $(t,r)\in\mathcal{T}_{\mathcal{I}\times\mathcal{K}}$, Lemma 5.11 states that adding $\alpha\widehat{A}B_{Y,b}$ to $Z|_{\hat{t}\times\hat{r}}$ takes

$$W_{\mathrm{up}}(t,r)\le C_{\mathrm{up}}\hat{k}^2B_{\mathcal{I}\times\mathcal{K}}(t,r) \text{ operations,}$$

giving us a total of not more than

$$C_{\mathrm{mml}}\hat{k}^2\left(B_{\mathcal{I}\times\mathcal{J}}(t,s)+B_{\mathcal{I}\times\mathcal{K}}(t,r)\right) \text{ operations.}$$

If $(t,r)\notin\mathcal{T}_{\mathcal{I}\times\mathcal{K}}$, the low-rank matrix is added to an auxiliary low-rank matrix. Due to Corollary 5.10 and the special case in the definition (5.13), this takes not more than

$$C_{\mathrm{up}}\hat{k}^2(|\hat{t}|+|\hat{r}|)\le C_{\mathrm{mml}}\hat{k}^2B_{\mathcal{I}\times\mathcal{K}}(t,r) \text{ operations,}$$

and we obtain the same estimate as before.

Case 1b: If $b:=(s,r)\in\mathcal{L}^{-}_{\mathcal{J}\times\mathcal{K}}$, the admissibility of the block tree implies $s\in\mathcal{L}_{\mathcal{J}}$ or $r\in\mathcal{L}_{\mathcal{K}}$. In the first case, we have $|\hat{s}|\le r_{\mathcal{J}}\le\hat{k}$, in the second case $|\hat{r}|\le r_{\mathcal{K}}\le\hat{k}$, so we can conclude $\min\{|\hat{s}|,|\hat{r}|\}\le\hat{k}$.

If now $|\hat{s}|\le|\hat{r}|$ holds, Theorem 5.6 states that the call to `addeval_hmatrix` requires not more than

$$W_{\mathrm{mv}}(t,s,\hat{s})\le 2\hat{k}^2B_{\mathcal{I}\times\mathcal{J}}(t,s) \text{ operations.}$$

Otherwise, the call requires not more than

$$W_{\mathrm{mv}}(t,s,\hat{r})\le 2\hat{k}^2B_{\mathcal{I}\times\mathcal{J}}(t,s) \text{ operations.}$$

In both cases, we obtain a factorized low-rank representation with a rank bounded by $\min\{|\hat{s}|, |\hat{r}|\} \leq \hat{k}$, and Corollary 5.10 and Lemma 5.11 yield that the call to `add_rkmatrix_hmatrix` requires not more than

$$W_{\mathrm{up}}(t, r) \leq C_{\mathrm{up}} \hat{k}^2 B_{\mathcal{I}\times\mathcal{K}}(t, r) \text{ operations,}$$

which brings the total to not more than

$$C_{\mathrm{mml}} \hat{k}^2 \left(B_{\mathcal{I}\times\mathcal{J}}(t, s) + B_{\mathcal{I}\times\mathcal{K}}(t, r)\right) \text{ operations.}$$

Case 2a: If $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, we compute $\widehat{B} = Y|^*_{\hat{s}\times\hat{r}} B_{X,(t,s)}$ by using the function `addevaltrans_hmatrix` for the matrix $B_{X,(t,s)}$ with k columns. Theorem 5.6 yields that this takes not more than

$$W_{\mathrm{mv}}(s, r, [1 : k]) \leq 2\hat{k}^2 B_{\mathcal{J}\times\mathcal{K}}(s, r) \text{ operations.}$$

As in Case 1a, we can use Corollary 5.10 and Lemma 5.11 to see that adding the resulting low-rank matrix to $Z|_{\hat{t}\times\hat{r}}$ requires not more than

$$C_{\mathrm{up}} \hat{k}^2 B_{\mathcal{I}\times\mathcal{K}}(t, r) \text{ operations,}$$

for a total of

$$2\hat{k}^2 B_{\mathcal{J}\times\mathcal{K}}(s, r) + C_{\mathrm{up}} \hat{k}^2 B_{\mathcal{I}\times\mathcal{K}}(t, r) \leq C_{\mathrm{mml}} \hat{k}^2 (B_{\mathcal{J}\times\mathcal{K}}(s, r) + B_{\mathcal{I}\times\mathcal{K}}(t, r)).$$

Case 2b: If $b = (t, s) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$, the admissibility of the block tree implies $t \in \mathcal{L}_{\mathcal{I}}$ or $s \in \mathcal{L}_{\mathcal{J}}$, and as in Case 1b we obtain $\min\{|\hat{t}|, |\hat{s}|\} \leq \hat{k}$.

If $|\hat{s}| \leq |\hat{t}|$, Theorem 5.6 states that the call to `addevaltrans_hmatrix` takes not more than

$$W_{\mathrm{mv}}(s, r, \hat{s}) \leq 2\hat{k}^2 B_{\mathcal{J}\times\mathcal{K}}(s, r) \text{ operations.}$$

Otherwise, the call requires not more than

$$W_{\mathrm{mv}}(s, r, \hat{t}) \leq 2\hat{k}^2 B_{\mathcal{J}\times\mathcal{K}}(s, r) \text{ operations.}$$

In both cases, the rank of the resulting low-rank representation is bounded by $\min\{|\hat{t}|, |\hat{s}|\} \leq \hat{k}$, and Corollary 5.10 and Lemma 5.11 yield that the call to the function `add_rkmatrix_hmatrix` requires not more than

$$C_{\mathrm{up}} \hat{k}^2 B_{\mathcal{I}\times\mathcal{K}}(t, r) \text{ operations,}$$

for a total of

$$2\hat{k}^2 B_{\mathcal{J}\times\mathcal{K}}(s, r) + C_{\mathrm{up}} \hat{k}^2 B_{\mathcal{I}\times\mathcal{K}}(t, r) \leq C_{\mathrm{mml}} \hat{k}^2 (B_{\mathcal{J}\times\mathcal{K}}(s, r) + B_{\mathcal{I}\times\mathcal{K}}(t, r)).$$

This covers all leaf cases and the proof is complete. ■

Theorem 5.18 (Complexity) *For all $(t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$, we define*

$$W_{mm}(t,s,r) := \begin{cases} C_{mml}\hat{k}^2(B_{\mathcal{I}\times\mathcal{J}}(t,s) + B_{\mathcal{J}\times\mathcal{K}}(s,r) + B_{\mathcal{I}\times\mathcal{K}}(t,r)) & \text{if } \operatorname{chil}(t,s,r) = \emptyset, \\ 2C_{mg}\hat{k}^2\sigma^3(|\hat{t}| + |\hat{r}|) + \sum\limits_{(t',s',r')\in\operatorname{chil}(t,s,r)} W_{mm}(t',s',r') & \text{otherwise.} \end{cases}$$

If we call the algorithm **addmul_hmatrix** *with $(t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$, it requires not more than $W_{mm}(t,s,r)$ operations.*

We have

$$W_{mm}(t,s,r) \le C_{mm}C_{\mathrm{sp}}^2\hat{k}^2(\hat{p}+1)^2(|\hat{t}| + |\hat{s}| + |\hat{r}|) \qquad \text{for all } (t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$$

with $C_{mm} := 3\max\{C_{mml}, 2C_{mg}\sigma^3\}$.

Proof. We denote the subtree of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ for the root (t,s,r) (cf. Lemma 5.3) by $\mathcal{T}_{(t,s,r)}$ and prove the first part by induction over $|\mathcal{T}_{(t,s,r)}|$.

If we have $|\mathcal{T}_{(t,s,r)}| = 1$, (t,s,r) has to be a leaf of the product tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$, and Lemma 5.17 yields the required bound.

Let now $n \in \mathbb{N}$ be such that the complexity bound holds for all $(t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ with $|\mathcal{T}_{(t,s,r)}| \le n$.

Let $(t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ with $|\mathcal{T}_{(t,s,r)}| = n+1$. The triple (t,s,r) cannot be a leaf of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$, so the algorithm addmul_hmatrix calls itself recursively for all $(t',s',r') \in \operatorname{chil}(t,s,r)$. For each of these triples $(t',s',r') \in \operatorname{chil}(t,s,r)$, we have $|\mathcal{T}_{(t',s',r')}| \le n$ and can apply the induction assumption to find that the recursive call requires not more than $W_{\mathrm{mm}}(t',s',r')$ operations.

If we have $(t,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{K}} \setminus \mathcal{L}_{\mathcal{I}\times\mathcal{K}}$, no additional operations are required. If $(t,r) \notin \mathcal{T}_{\mathcal{I}\times\mathcal{K}} \setminus \mathcal{L}_{\mathcal{I}\times\mathcal{K}}$, the auxiliary submatrices created by our algorithm have to be merged using rowmerge_rkmatrix and colmerge_rkmatrix. According to Lemma 5.12, this takes not more than

$$\begin{aligned} C_{\mathrm{mg}}\sigma^2\hat{k}^2(|\hat{t}| + |\hat{r}|) &+ \sum_{t'\in\operatorname{chil}^+(t)} C_{\mathrm{mg}}\sigma^2\hat{k}^2(|\hat{t}'| + |\hat{r}|) \\ &\le C_{\mathrm{mg}}\sigma^2\hat{k}^2(|\hat{t}| + |\hat{r}|) + C_{\mathrm{mg}}\sigma^2\hat{k}^2|\hat{t}| + C_{\mathrm{mg}}\sigma^3\hat{k}^2|\hat{r}| \\ &\le 2C_{\mathrm{mg}}\sigma^3\hat{k}^2(|\hat{t}| + |\hat{r}|) \end{aligned}$$

operations if we merge the rows (corresponding to the children of t) first, followed by the columns. This completes the proof of the first part.

To prove the second part, we introduce $\widehat{C}_{\mathrm{mm}} := \max\{C_{\mathrm{mml}}, 2C_{\mathrm{mg}}\sigma^3\}$. A straightforward induction using

$$|\hat{t}| + |\hat{r}| \le B_{\mathcal{I}\times\mathcal{K}}(t,r) \qquad\qquad \text{for all } (t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$$

yields the estimate

$$W_{\mathrm{mm}}(t,s,r) \le \widehat{C}_{\mathrm{mm}}\hat{k}^2 \sum_{(t',s',r')\in\mathcal{T}_{(t,s,r)}} B_{\mathcal{I}\times\mathcal{J}}(t',s') + B_{\mathcal{J}\times\mathcal{K}}(s',r') + B_{\mathcal{I}\times\mathcal{K}}(t',r')$$

for all $(t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$.

Let $(t,s,r) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$. Due to Lemma 5.16, we have

$$\begin{aligned}
\sum_{(t',s',r')\in\mathcal{T}_{(t,s,r)}} B_{\mathcal{I}\times\mathcal{J}}(t',s') &\le \sum_{(t',s')\in\mathcal{T}_{(t,s)}} \sum_{\substack{r'\in\mathcal{T}_{\mathcal{K}}\\(s',r')\in\mathcal{T}_{(s,r)}}} B_{\mathcal{I}\times\mathcal{J}}(t',s')\\
&\le C_{\text{sp}} \sum_{(t',s')\in\mathcal{T}_{(t,s)}} B_{\mathcal{I}\times\mathcal{J}}(t',s')\\
&= C_{\text{sp}} \sum_{(t',s')\in\mathcal{T}_{(t,s)}} \sum_{(t'',s'')\in\mathcal{T}_{(t',s')}} |\hat{t}''| + |\hat{s}''|\\
&\le C_{\text{sp}} \sum_{(t'',s'')\in\mathcal{T}_{(t,s)}} \sum_{(t',s')\in\operatorname{pred}(t'',s'')} |\hat{t}''| + |\hat{s}''|\\
&\le C_{\text{sp}}(p_{\mathcal{I}\times\mathcal{J}}+1) \sum_{(t'',s'')\in\mathcal{T}_{(t,s)}} |\hat{t}''| + |\hat{s}''|.
\end{aligned}$$

Now we can apply Lemma 3.34 to conclude

$$\sum_{(t',s',r')\in\mathcal{T}_{(t,s,r)}} B_{\mathcal{I}\times\mathcal{J}}(t',s') \le C_{\text{sp}}^2(p_{\mathcal{I}\times\mathcal{J}}+1)^2(|\hat{t}|+|\hat{s}|). \tag{5.14a}$$

We can use the same arguments to find

$$\sum_{(t',s',r')\in\mathcal{T}_{(t,s,r)}} B_{\mathcal{J}\times\mathcal{K}}(s',r') \le C_{\text{sp}}^2(p_{\mathcal{J}\times\mathcal{K}}+1)^2(|\hat{s}|+|\hat{r}|). \tag{5.14b}$$

For the third term, we have to distinguish between the triples $(t',s',r') \in \mathcal{T}_{(t,s,r)}$ with $(t',r') \in \mathcal{T}_{\mathcal{I}\times\mathcal{K}}$ and the remainder. Using Lemma 5.16, we obtain

$$\begin{aligned}
\sum_{\substack{(t',s',r')\in\mathcal{T}_{(t,s,r)}\\(t',r')\in\mathcal{T}_{\mathcal{I}\times\mathcal{K}}}} B_{\mathcal{I}\times\mathcal{K}}(t',r') &\le \sum_{(t',r')\in\mathcal{T}_{\mathcal{I}\times\mathcal{K}}} \sum_{\substack{s'\in\mathcal{T}_{\mathcal{J}}\\(t',s')\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}}}} B_{\mathcal{I}\times\mathcal{K}}(t',r')\\
&\le C_{\text{sp}} \sum_{(t',r')\in\mathcal{T}_{\mathcal{I}\times\mathcal{K}}} B_{\mathcal{I}\times\mathcal{K}}(t',r')
\end{aligned}$$

and can proceed as before to get

$$\sum_{\substack{(t',s',r')\in\mathcal{T}_{(t,s,r)}\\(t',r')\in\mathcal{T}_{\mathcal{I}\times\mathcal{K}}}} B_{\mathcal{I}\times\mathcal{K}}(t',r') \le C_{\text{sp}}^2(p_{\mathcal{I}\times\mathcal{K}}+1)^2(|\hat{t}|+|\hat{r}|). \tag{5.14c}$$

If we have $(t',s',r') \in \mathcal{T}_{(t,s,r)}$ with $(t',r') \notin \mathcal{T}_{\mathcal{I}\times\mathcal{K}}$, our definition implies $B_{\mathcal{I}\times\mathcal{K}}(t',r') = |\hat{t}'| + |\hat{r}'|$ and we can use Lemma 5.16 again to find

$$\sum_{\substack{(t',s',r')\in\mathcal{T}_{(t,s,r)}\\(t',r')\notin\mathcal{T}_{\mathcal{I}\times\mathcal{K}}}} B_{\mathcal{I}\times\mathcal{K}}(t',r') = \sum_{\substack{(t',s',r')\in\mathcal{T}_{(t,s,r)}\\(t',r')\notin\mathcal{T}_{\mathcal{I}\times\mathcal{K}}}} |\hat{t}'| + |\hat{r}'|$$

$$\leq \sum_{(t',s')\in\mathcal{T}_{(t,s)}} \sum_{\substack{r'\in\mathcal{T}_{\mathcal{K}}\\(s',r')\in\mathcal{T}_{(s,r)}}} |\hat{t}'| + \sum_{(s',r')\in\mathcal{T}_{(s,r)}} \sum_{\substack{t'\in\mathcal{T}_{\mathcal{I}}\\(t',s')\in\mathcal{T}_{(t,s)}}} |\hat{r}'|$$

$$\leq C_{\mathrm{sp}} \sum_{(t',s')\in\mathcal{T}_{(t,s)}} |\hat{t}'| + C_{\mathrm{sp}} \sum_{(s',r')\in\mathcal{T}_{(s,r)}} |\hat{r}'|.$$

Once again we can use Lemma 3.34 to obtain

$$\sum_{\substack{(t',s',r')\in\mathcal{T}_{(t,s,r)}\\(t',r')\not\in\mathcal{T}_{\mathcal{I}\times\mathcal{K}}}} B_{\mathcal{I}\times\mathcal{K}}(t',r') \leq C_{\mathrm{sp}}^2(p_{\mathcal{I}\times\mathcal{J}}+1)|\hat{t}| + C_{\mathrm{sp}}^2(p_{\mathcal{J}\times\mathcal{K}}+1)|\hat{r}|$$
$$\leq C_{\mathrm{sp}}^2(\hat{p}+1)(|\hat{t}|+|\hat{r}|). \tag{5.14d}$$

Accumulating the estimates (5.14a), (5.14b), (5.14c) and (5.14d) yields

$$\begin{aligned} W_{\mathrm{mm}}(t,s,r) &\leq \widehat{C}_{\mathrm{mm}} \sum_{(t',s',r')\in\mathcal{T}_{(t,s,r)}} B_{\mathcal{I}\times\mathcal{J}}(t',s') + B_{\mathcal{J}\times\mathcal{K}}(s',r') + B_{\mathcal{I}\times\mathcal{K}}(t',r') \\ &\leq \widehat{C}_{\mathrm{mm}} C_{\mathrm{sp}}^2(\hat{p}+1)^2(|\hat{t}|+|\hat{s}|+|\hat{s}|+|\hat{r}|+|\hat{t}|+|\hat{r}|+|\hat{t}|+|\hat{r}|) \\ &\leq 3\widehat{C}_{\mathrm{mm}} C_{\mathrm{sp}}^2(\hat{p}+1)^2(|\hat{t}|+|\hat{s}|+|\hat{r}|). \end{aligned}$$

Due to $C_{\mathrm{mm}} = 3\widehat{C}_{\mathrm{mm}}$, this is the required result. ■

Exercise 5.19 (Adjoint matrices) *Develop algorithms for efficiently performing the updates*

$$\begin{aligned} Z|_{\hat{t}\times\hat{r}} &\leftarrow Z|_{\hat{t}\times\hat{r}} + \alpha X|_{\hat{s}\times\hat{t}}^{*} Y|_{\hat{s}\times\hat{r}}, \\ Z|_{\hat{t}\times\hat{r}} &\leftarrow Z|_{\hat{t}\times\hat{r}} + \alpha X|_{\hat{t}\times\hat{s}} Y|_{\hat{r}\times\hat{s}}^{*}, \\ Z|_{\hat{t}\times\hat{r}} &\leftarrow Z|_{\hat{t}\times\hat{r}} + \alpha X|_{\hat{s}\times\hat{t}}^{*} Y|_{\hat{r}\times\hat{s}}^{*} \end{aligned}$$

for $(t,s,r)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ and approximating the results by $\mathcal{H}$-matrices.

In the first and third case, X is an $\mathcal{H}$-matrix for the adjoint block tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}^{*}$ constructed from $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ by swapping row and column clusters.

In the second and third case, Y is an $\mathcal{H}$-matrix for the adjoint block tree $\mathcal{T}_{\mathcal{J}\times\mathcal{K}}^{*}$ constructed from $\mathcal{T}_{\mathcal{J}\times\mathcal{K}}$ by swapping row and column clusters.

Exercise 5.20 (Cluster tree) *Prove that the product tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}$ is a cluster tree for the index set $\mathcal{I}\times\mathcal{J}\times\mathcal{K}$.*

Exercise 5.21 (Sparsity) *Assuming that $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ and $\mathcal{T}_{\mathcal{J}\times\mathcal{K}}$ are C_{sp}-sparse, prove*

$$\begin{aligned} |\{(s,r)\in\mathcal{T}_{\mathcal{J}}\times\mathcal{T}_{\mathcal{K}} \;:\; (t,s,r)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}\}| &\leq C_{\mathrm{sp}}^2 && \textit{for all } t\in\mathcal{T}_{\mathcal{I}}, \\ |\{(t,r)\in\mathcal{T}_{\mathcal{J}}\times\mathcal{T}_{\mathcal{K}} \;:\; (t,s,r)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}\}| &\leq C_{\mathrm{sp}}^2 && \textit{for all } s\in\mathcal{T}_{\mathcal{J}}, \\ |\{(t,s)\in\mathcal{T}_{\mathcal{J}}\times\mathcal{T}_{\mathcal{K}} \;:\; (t,s,r)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}\times\mathcal{K}}\}| &\leq C_{\mathrm{sp}}^2 && \textit{for all } r\in\mathcal{T}_{\mathcal{K}}. \end{aligned}$$

5.8 Inversion

With an efficient algorithm for the approximation of the product of two matrices at our disposal, we can consider another important algebraic operation: the inversion of a matrix, i.e., the construction of G^{-1} for a given $\mathcal{H}$-matrix G.

As in the case of the multiplication, we consider the more general problem of inverting a square submatrix $G|_{\hat{t}\times\hat{t}}$, since this allows us to formulate a recursive algorithm.

In this chapter and the next, we assume

- that $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{I}}$ is an $\mathcal{H}$-matrix for the C_{sp}-sparse admissible block tree $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$,
- that $G|_{\mathcal{M}\times\mathcal{M}}$ is invertible for all subsets $\mathcal{M} \subseteq \mathcal{I}$, and
- that for any $t \in \mathcal{L}_{\mathcal{I}}$, we have $(t,t) \in \mathcal{L}^{-}_{\mathcal{I}\times\mathcal{I}}$.

The third condition is required to ensure that we can compute the inverse of small diagonal blocks by the usual algorithms.

Case 1: t is a leaf. In this case (t,t) has to be an inadmissible leaf of $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$, i.e., we have $G|_{\hat{t}\times\hat{t}} = N_b$ for $b = (t,t) \in \mathcal{L}_{\mathcal{I}\times\mathcal{I}}$. We enumerate the indices, i.e., we have $\hat{t} = \{i_1, \ldots, i_n\}$ with $n := |\hat{t}|$, and define

$$g_{\nu\mu} := G_{i_\nu,i_\mu} \qquad\qquad \text{for all } \nu,\mu \in [1:n]$$

so that we can write the matrix in the usual form

$$G|_{\hat{t}\times\hat{t}} = \begin{pmatrix} g_{11} & \cdots & g_{1n} \\ \vdots & \ddots & \vdots \\ g_{n1} & \cdots & g_{nn} \end{pmatrix}.$$

In order to construct the inverse by recursion, we have to be able to reduce the dimension. We introduce

$$G_{1*} := \begin{pmatrix} g_{12} & \cdots & g_{1n} \end{pmatrix}, \qquad G_{*1} := \begin{pmatrix} g_{21} \\ \vdots \\ g_{n1} \end{pmatrix}, \qquad G_{**} := \begin{pmatrix} g_{22} & \cdots & g_{2n} \\ \vdots & \ddots & \vdots \\ g_{n2} & \cdots & g_{nn} \end{pmatrix}$$

and find

$$G|_{\hat{t}\times\hat{t}} = \begin{pmatrix} g_{11} & G_{1*} \\ G_{*1} & G_{**} \end{pmatrix}.$$

Assuming $g_{11} \neq 0$, a partial LR factorization is given by

$$G|_{\hat{t}\times\hat{t}} = \begin{pmatrix} 1 & \\ G_{*1}g_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} g_{11} & G_{1*} \\ & G_{**} - G_{*1}g_{11}^{-1}G_{1*} \end{pmatrix}.$$

The inverse can be obtained by inverting both triangular factors and multiplying them in reversed order. For the lower triangular factor, we have

$$\begin{pmatrix} 1 & \\ G_{*1}g_{11}^{-1} & I \end{pmatrix}^{-1} = \begin{pmatrix} 1 & \\ -G_{*1}g_{11}^{-1} & I \end{pmatrix}.$$

Introducing the *Schur complement* $S := G_{**} - G_{*1} g_{11}^{-1} G_{1*}$, we find

$$\begin{pmatrix} g_{11} & G_{1*} \\ & S \end{pmatrix}^{-1} = \begin{pmatrix} g_{11}^{-1} & -g_{11}^{-1} G_{1*} S^{-1} \\ & S^{-1} \end{pmatrix},$$

and the inverse is given by

$$\begin{aligned} G|_{\hat{t}\times\hat{t}}^{-1} &= \begin{pmatrix} g_{11} & G_{1*} \\ & S \end{pmatrix}^{-1} \begin{pmatrix} 1 & \\ G_{*1} g_{11}^{-1} & I \end{pmatrix}^{-1} \\ &= \begin{pmatrix} g_{11}^{-1} & -g_{11}^{-1} G_{1*} S^{-1} \\ & S^{-1} \end{pmatrix} \begin{pmatrix} 1 & \\ -G_{*1} g_{11}^{-1} & I \end{pmatrix} \\ &= \begin{pmatrix} g_{11}^{-1} + g_{11}^{-1} G_{1*} S^{-1} G_{*1} g_{11}^{-1} & -g_{11} G_{1*} S^{-1} \\ -S^{-1} G_{*1} g_{11}^{-1} & S^{-1} \end{pmatrix}. \end{aligned}$$

The entire computation can be split into eight steps:

1. Invert g_{11}.
2. Compute $H_{*1} := G_{*1} g_{11}^{-1}$.
3. Compute $H_{1*} := g_{11}^{-1} G_{1*}$.
4. Compute $S := G_{**} - G_{*1} g_{11}^{-1} G_{1*} = G_{**} - H_{*1} G_{1*}$.
5. Invert S, and let $Z_{**} := S^{-1}$.
6. Compute $Z_{*1} := -S^{-1} G_{*1} g_{11}^{-1} = -S^{-1} H_{*1}$.
7. Compute $Z_{1*} := -g_{11}^{-1} G_{1*} S^{-1} = -H_{1*} S^{-1}$.
8. Compute $z_{11} := g_{11}^{-1} + g_{11}^{-1} G_{1*} S^{-1} G_{*1} g_{11}^{-1} = g_{11}^{-1} - H_{1*} Z_{*1}$.

The inverse is then given by

$$G|_{\hat{t}\times\hat{t}}^{-1} = \begin{pmatrix} z_{11} & Z_{1*} \\ Z_{*1} & Z_{**} \end{pmatrix}$$

We can see that all steps except for the first and the fifth require only multiplications of submatrices, while the first and fifth step require us to invert submatrices. The first step is trivial, the fifth can be handled by recursion.

Since the submatrices G_{**}, S and S^{-1} appearing in this approach are always of the form

$$\begin{pmatrix} g_{kk} & \cdots & g_{kn} \\ \vdots & \ddots & \vdots \\ g_{nk} & \cdots & g_{nn} \end{pmatrix},$$

we can avoid using an explicit recursion by using a loop that runs over $k = 1, 2, \ldots, n$ and performs the steps 1 to 4 for the corresponding submatrices, and a second loop that runs over $k = n, n-1, \ldots, 1$ and performs the steps 5 to 8, computing the inverses of the lower right submatrices.

The resulting algorithm is given in Figure 5.9.

```
procedure invert_amatrix(var $G$, $H$);
  for $k = 1$ to $n$ do begin
    $g_{kk} \leftarrow g_{kk}^{-1}$;
    for $i \in [k+1:n]$ do $h_{ik} \leftarrow g_{ik} g_{kk}$;
    for $i \in [k+1:n]$ do $h_{ki} \leftarrow g_{kk} g_{ki}$;
    for $i, j \in [k+1:n]$ do $g_{ij} \leftarrow g_{ij} - h_{ik} g_{kj}$
  end;
  for $k = n$ downto $1$ do begin
    for $i \in [k+1:n]$ do begin
      $g_{ik} \leftarrow 0$;
      for $j \in [k+1:n]$ do $g_{ik} \leftarrow g_{ik} - g_{ij} h_{jk}$;
      $g_{ki} \leftarrow 0$;
      for $j \in [k+1:n]$ do $g_{ki} \leftarrow g_{ki} - h_{kj} g_{ji}$
    end;
    for $i \in [k+1:n]$ do $g_{kk} \leftarrow g_{kk} - h_{ki} g_{ik}$
  end
```

Figure 5.9: Inversion of a matrix in standard array representation.

Lemma 5.22 (Complexity) *The algorithm* `invert_amatrix` *takes* $2n^3 - n^2$ *operations to compute the inverse of an* $n \times n$ *matrix.*

Proof. The algorithm takes

$$\begin{aligned}\sum_{k=1}^{n} 1 + 4(n-k) + 6(n-k)^2 &= n + 4\sum_{\ell=0}^{n-1} \ell + 6\sum_{\ell=0}^{n-1} \ell^2 \\ &= n + 4\frac{n(n-1)}{2} + 6\frac{n(n-1)(2n-1)}{6} \\ &= n + 2n(n-1) + n(n-1)(2n-1) \\ &= n + n(n-1)(2n+1) \\ &= n + 2n^2(n-1) + n(n-1) = 2n^2(n-1) + n^2 \\ &= 2n^3 - n^2 \text{ operations.}\end{aligned}$$

∎

We note that the inversion takes *exactly* the same number of operations as the straightforward computation of $G|_{\hat{t}\times\hat{t}}^2$, although in practice divisions take far longer than multiplications on modern processors.

Case 2: t is not a leaf. In this case, $G|_{\hat{t}\times\hat{t}}$ has to be subdivided, i.e., we have $\mathrm{chil}(b) = \mathrm{chil}(t) \times \mathrm{chil}(t)$. We proceed as in the first case, but replace individual indices by child clusters: we let $\{t_1, \ldots, t_n\} := \mathrm{chil}(t)$ with $n := |\,\mathrm{chil}(t)|$ and define submatrices

$$G_{\nu\mu} := G|_{\hat{t}_\nu \times \hat{t}_\mu} \qquad \text{for all } \nu, \mu \in [1:n],$$

so that we have the block matrix representation

$$G|_{\hat{t}\times\hat{t}} = \begin{pmatrix} G_{11} & \dots & G_{1n} \\ \vdots & \ddots & \vdots \\ G_{n1} & \dots & G_{nn} \end{pmatrix}.$$

We define submatrices

$$G_{1*} := \begin{pmatrix} G_{12} & \dots & G_{1n} \end{pmatrix}, \qquad G_{*1} := \begin{pmatrix} G_{21} \\ \vdots \\ G_{n1} \end{pmatrix}, \qquad G_{**} := \begin{pmatrix} G_{22} & \dots & G_{2n} \\ \vdots & \ddots & \vdots \\ G_{n2} & \dots & G_{nn} \end{pmatrix},$$

assume that G_{11} is invertible to introduce the partial block LR factorization

$$G|_{\hat{t}\times\hat{t}} = \begin{pmatrix} I & \\ G_{*1}G_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} G_{11} & G_{1*} \\ & S \end{pmatrix}, \qquad S := G_{**} - G_{*1}G_{11}^{-1}G_{1*},$$

and obtain the representation

$$\begin{aligned} G|_{\hat{t}\times\hat{t}}^{-1} &= \begin{pmatrix} G_{11} & G_{1*} \\ & S \end{pmatrix}^{-1} \begin{pmatrix} I & \\ G_{*1}G_{11}^{-1} & I \end{pmatrix}^{-1} = \begin{pmatrix} G_{11}^{-1} & -G_{11}^{-1}G_{1*}S^{-1} \\ & S^{-1} \end{pmatrix} \begin{pmatrix} I & \\ -G_{*1}G_{11}^{-1} & I \end{pmatrix} \\ &= \begin{pmatrix} G_{11}^{-1} + G_{11}^{-1}G_{1*}S^{-1}G_{*1}G_{11}^{-1} & -G_{11}^{-1}G_{1*}S^{-1} \\ -S^{-1}G_{*1}G_{11}^{-1} & S^{-1} \end{pmatrix} \end{aligned}$$

for the inverse of $G|_{\hat{t}\times\hat{t}}$. We can compute this matrix by exactly the same procedure as before, as long as we replace individual coefficients by submatrices and multiplication and inversion of submatrices by the corresponding approximative $\mathcal{H}$-matrix operations. The resulting algorithm is given in Figure 5.10.

Our goal is now to obtain an upper bound for the number of operations requires by `invert_hmatrix`. We can follow the same approach as for the matrix multiplication, i.e., we can start with an inductively defined bound inspired by the recursive structure of the algorithm.

Lemma 5.23 (Complexity bound) *Let $t \in \mathcal{T}_{\mathcal{I}}$. The number of operations required by the function `invert_hmatrix` to approximate the inverse is bounded by*

$$W_{inv}(t) := \begin{cases} 2|\hat{t}|^3 - |\hat{t}|^2 & \text{if } \operatorname{chil}(t) = \emptyset, \\ \sum_{t' \in \operatorname{chil}(t)} W_{inv}(t') + \sum_{\substack{t',s',r' \in \operatorname{chil}(t) \\ t' \neq s' \vee t' \neq r'}} W_{mm}(t',s',r') & \text{otherwise,} \end{cases}$$

for all $t \in \mathcal{T}_{\mathcal{I}}$.

Proof. Let $t \in \mathcal{T}_{\mathcal{I}}$. If $\operatorname{chil}(t) = \emptyset$, the call to `invert_amatrix` requires

$$2|\hat{t}|^3 - |\hat{t}|^2 = W_{\text{inv}}(t) \texttt{ operations}$$

according to Lemma 5.22.

```
procedure invert_hmatrix($t$, var $G$, $H$);
  if $\operatorname{chil}(t) = \emptyset$ then
    invert_amatrix($t$, $N_{G,(t,t)}$, $N_{H,(t,t)}$);
  else begin
    $n \leftarrow |\operatorname{chil}(t)|$;    $\{t_1, \ldots, t_n\} \leftarrow \operatorname{chil}(t)$;
    for $k = 1$ to $n$ do begin
      invert_hmatrix($t_k$, $G$, $H$);
      for $i \in [k+1:n]$ do
        zero_hmatrix($t_i$, $t_k$, $H$);    addmul_hmatrix($t_i$, $t_k$, $t_k$, 1, $G$, $G$, $H$);
      for $i \in [k+1:n]$ do
        zero_hmatrix($t_k$, $t_i$, $H$);    addmul_hmatrix($t_k$, $t_k$, $t_i$, 1, $G$, $G$, $H$);
      for $i, j \in [k+1:n]$ do addmul_hmatrix($t_i$, $t_k$, $t_j$, $-1$, $H$, $G$, $G$)
    end;
    for $k = n$ downto 1 do begin
      for $i \in [k+1:n]$ do begin
        zero_hmatrix($t_i$, $t_k$, $G$);
        for $j \in [k+1:n]$ do addmul_hmatrix($t_i$, $t_j$, $t_k$, -1, $G$, $H$, $G$);
        zero_hmatrix($t_k$, $t_i$, $G$);
        for $j \in [k+1:n]$ do addmul_hmatrix($t_k$, $t_j$, $t_i$, -1, $H$, $G$, $G$)
      end;
      for $j \in [k+1:n]$ do addmul_hmatrix($t_k$, $t_j$, $t_k$, -1, $H$, $G$, $G$)
    end
  end
```

Figure 5.10: Approximative inversion of an $\mathcal{H}$-matrix.

If $\operatorname{chil}(t) \neq \emptyset$, we let $n := |\operatorname{chil}(t)|$ and $\{t_1, \ldots, t_n\} := \operatorname{chil}(t)$. We call the function `invert_hmatrix` recursively for t_k with $k \in [1:n]$, and this takes

$$\sum_{k=1}^{n} W_{\text{inv}}(t_k) = \sum_{t' \in \operatorname{chil}(t)} W_{\text{inv}}(t') \text{ operations.}$$

The calls to `addmul_hmatrix` require

$$\begin{aligned}
&\sum_{k=1}^{n} \sum_{i=k+1}^{n} W_{\text{mm}}(t_i, t_k, t_k) + \sum_{k=1}^{n} \sum_{i=k+1}^{n} W_{\text{mm}}(t_k, t_k, t_i) \\
&\quad + \sum_{k=1}^{n} \sum_{i=k+1}^{n} \sum_{j=k+1}^{n} W_{\text{mm}}(t_i, t_k, t_j) + \sum_{k=1}^{n} \sum_{i=k+1}^{n} \sum_{j=k+1}^{n} W_{\text{mm}}(t_i, t_j, t_k) \\
&\quad + \sum_{k=1}^{n} \sum_{i=k+1}^{n} \sum_{j=k+1}^{n} W_{\text{mm}}(t_k, t_j, t_i) + \sum_{k=1}^{n} \sum_{i=k+1}^{n} W_{\text{mm}}(t_k, t_i, t_k) \text{ operations.}
\end{aligned}$$

The corresponding index sets are given by

$$\begin{aligned}
S_1 &:= \{(i,j,k) \;:\; i,j,k \in [1:n],\ j = k = \min\{i,j,k\},\ i > j\},\\
S_2 &:= \{(i,j,k) \;:\; i,j,k \in [1:n],\ i = j = \min\{i,j,k\},\ k > i\},\\
S_3 &:= \{(i,j,k) \;:\; i,j,k \in [1:n],\ j = \min\{i,j,k\},\ i,k > j\},\\
S_4 &:= \{(i,j,k) \;:\; i,j,k \in [1:n],\ k = \min\{i,j,k\},\ i,j > k\},\\
S_5 &:= \{(i,j,k) \;:\; i,j,k \in [1:n],\ i = \min\{i,j,k\},\ j,k > i\},\\
S_6 &:= \{(i,j,k) \;:\; i,j,k \in [1:n],\ i = k = \min\{i,j,k\},\ j > i\},
\end{aligned}$$

and they are disjoint due to the different positions the minimal index takes in the triples. In the sets S_1, S_2 and S_6, exactly two indices equal the minimum, in the sets S_3, S_4 and S_5, exactly one index equals the minimum, so the only missing combinations are the ones where all three indices equal the minimum This observation implies

$$S_1 \cup S_2 \cup S_3 \cup S_4 \cup S_5 \cup S_6 = \{(i,j,k) \;:\; i,j,k \in [1:n],\ i \neq j \vee i \neq k\},$$

and we conclude

$$W_{\mathrm{inv}}(t) = \sum_{t' \in \mathrm{chil}(t)} W_{\mathrm{inv}}(t') + \sum_{\substack{t',s',r' \in \mathrm{chil}(t)\\ t' \neq s' \vee t' \neq r'}} W_{\mathrm{mm}}(t',s',r').$$

∎

Theorem 5.24 (Complexity) *We have*

$$W_{inv}(t) \le W_{mm}(t,t,t) \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}}.$$

Proof. By induction on $|\mathcal{T}_t|$.

Let $t \in \mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t| = 1$. Then we have $\mathrm{chil}(t) = \emptyset$ and

$$W_{\mathrm{inv}}(t) = 2|\hat{t}|^3 - |\hat{t}|^2 < 2|\hat{t}|^3 \le C_{\mathrm{mml}}\hat{k}^2|\hat{t}| \le W_{\mathrm{mm}}(t,t,t).$$

Let now $m \in \mathbb{N}$ be such that the inequality holds for all $t \in \mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t| \le m$. Let $t \in \mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t| = m+1$. Then we have $\mathrm{chil}(t) \neq \emptyset$ and

$$W_{\mathrm{inv}}(t) = \sum_{t' \in \mathrm{chil}(t)} W_{\mathrm{inv}}(t') + \sum_{\substack{t',s',r' \in \mathrm{chil}(t)\\ s' \neq t' \vee r' \neq t'}} W_{\mathrm{mm}}(t',s',r') \text{ operations.}$$

For all $t' \in \mathrm{chil}(t)$, $t \notin \mathcal{T}_{t'}$ implies $|\mathcal{T}_{t'}| \le m$, so we can apply the induction assumption to find

$$W_{\mathrm{inv}}(t') \le W_{\mathrm{mm}}(t',t',t') \qquad \text{for all } t' \in \mathrm{chil}(t).$$

This yields

$$W_{\mathrm{inv}}(t) \le \sum_{t',s',r' \in \mathrm{chil}(t)} W_{\mathrm{mm}}(t',s',r') \le W_{\mathrm{mm}}(t,t,t).$$

∎

Corollary 5.25 (Complexity) *The approximate inversion of an $\mathcal{H}$-matrix requires not more than*

$$6C_{\mathrm{sp}}^2 C_{mm} \hat{k}^2 (\hat{p}+1)^2 |\mathcal{I}| \text{ operations.}$$

Proof. Combine Theorem 5.24 with Theorem 5.18. ■

5.9 Triangular factorizations

Frequently, computing the inverse of a matrix is not really necessary, since a factorization can be used instead. We consider the LR factorization as a typical example, i.e., the factorization of a matrix $G = LR$ into a left lower triangular matrix L and a right upper triangular matrix R.

Before we consider the construction of a factorization, let us first investigate the corresponding procedure for solving a linear system $Gx = b$. If we have the lower and upper triangular matrices L and R with $G = LR$ at our disposal, we can introduce an auxiliary vector y and obtain

$$Ly = b, \qquad\qquad Rx = y,$$

so we only have to be able to solve triangular systems. The well-known forward and backward substitution algorithms handle this task very efficiently.

We have to assume that the index set $\mathcal{I}$ is totally ordered, since otherwise there is no useful way to even define triangular matrices. We also have to assume that the order of the index set is compatible with the cluster tree.

Definition 5.26 (Compatible order) *Let $\mathcal{T}_{\mathcal{I}}$ be a cluster tree for a totally ordered index set $\mathcal{I}$. The order of $\mathcal{I}$ is* compatible *with the cluster tree if*

$$(\exists i \in \hat{t}', j \in \hat{s}' \ : \ i \le j) \implies (\forall i \in \hat{t}', j \in \hat{s}' \ : \ i \le j) \quad \text{for all } t \in \mathcal{T}_{\mathcal{I}},\ t', s' \in \operatorname{chil}(t), \text{ with } t' \neq s',$$

i.e., if all indices corresponding to one child are either strictly greater or lesser than all indices corresponding to their siblings.

Remark 5.27 (Compatible order) *In practical implementations, a total order on $\mathcal{I}$ is constructed along with the cluster tree: if t is a leaf, we choose a total order for $\hat{t}$. If t is not a leaf, we choose a total order for* $\operatorname{chil}(t)$.

Given $i, j \in \mathcal{I}$ with $i \neq j$, we find the cluster $t \in \mathcal{T}_{\mathcal{I}}$ of maximal level such that $i, j \in \hat{t}$. If t is a leaf, we use the order given for leaf clusters. If t is not a leaf, there are children t' and s' with $i \in \hat{t}'$ and $j \in \hat{s}'$. Since we have chosen t to have the maximal level, we have $j \notin \hat{t}'$ and therefore $t' \neq s'$. We use the order given for the children: if $t' < s'$, we let $i \le j$, and we let $j \le i$ otherwise.

Unless we are dealing with one-dimensional problems, the "natural" order imposed by an implementation of the discretization will usually not be compatible with the cluster tree $\mathcal{T}_{\mathcal{I}}$, therefore we have to bear in mind that a triangular $\mathcal{H}$-matrix is triangular with respect to a compatible order, not necessarily the order used by the underlying application.

In the following, we assume that the order of $\mathcal{I}$ is compatible with $\mathcal{T}_{\mathcal{I}}$.

Solving triangular systems

Let now L be a left lower triangular $\mathcal{H}$-matrix. As in the case of the multiplication and the inversion, we consider the task of solving

$$L|_{\hat{t}\times\hat{t}} x = y \tag{5.15}$$

for a given cluster $t \in \mathcal{T}_{\mathcal{I}}$ and vectors $x, y \in \mathbb{K}^{\hat{t}}$.

Case 1: t is a leaf. In this case (t,t) has to be an inadmissible leaf of $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$, i.e., we have $L|_{\hat{t}\times\hat{t}} = N_b$ for $b = (t,t) \in \mathcal{L}_{\mathcal{I}\times\mathcal{I}}$. As in the case of the inversion algorithm, we let $n := |\hat{t}|$ and $\hat{t} = \{i_1, \ldots, i_n\}$. Since we have an order on $\hat{t}$ at our disposal, we ensure

$$\nu \leq \mu \Longrightarrow i_\nu \leq i_\mu \qquad\qquad \text{for all } \nu, \mu \in [1:n].$$

Using

$$\ell_{\nu\mu} := \ell_{i_\nu, i_\nu}, \qquad x_\mu := x_{i_\mu}, \qquad y_\nu := y_{i_\nu} \qquad \text{for all } \nu, \mu \in [1:n],$$

and taking advantage of the fact that L is left lower triangular, we have

$$L|_{\hat{t}\times\hat{t}} = \begin{pmatrix} \ell_{11} & & \\ \vdots & \ddots & \\ \ell_{n1} & \ldots & \ell_{nn} \end{pmatrix}, \qquad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \qquad y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

The forward substitution algorithm can be interpreted as a recursive procedure: we let

$$L_{*1} := \begin{pmatrix} \ell_{21} \\ \vdots \\ \ell_{n1} \end{pmatrix}, \quad L_{**} := \begin{pmatrix} \ell_{22} & & \\ \vdots & \ddots & \\ \ell_{n2} & \ldots & \ell_{nn} \end{pmatrix}, \quad x_* := \begin{pmatrix} x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad y_* := \begin{pmatrix} y_2 \\ \vdots \\ y_n \end{pmatrix}$$

and have that (5.15) is equivalent to

$$\begin{pmatrix} \ell_{11} & \\ L_{*1} & L_{**} \end{pmatrix} \begin{pmatrix} x_1 \\ x_* \end{pmatrix} = \begin{pmatrix} y_1 \\ y_* \end{pmatrix},$$

i.e., to the two equations

$$\ell_{11} x_1 = y_1, \qquad\qquad L_{**} x_* = y_* - L_{*1} x_1.$$

```
**procedure** lowersolve_amatrix($L$, **var** $X$);
  **for** $k = 1$ **to** $n$ **do begin**
    **for** $j \in \mathcal{M}$ **do** $x_{k,j} \leftarrow x_{k,j}/\ell_{kk}$;
    **for** $i \in [k+1:n]$, $j \in \mathcal{M}$ **do** $x_{i,j} \leftarrow x_{i,j} - \ell_{ik}x_{k,j}$
  **end**

**procedure** uppersolve_amatrix($R$, **var** $X$);
  **for** $k = n$ **downto** $1$ **do begin**
    **for** $j \in \mathcal{M}$ **do** $x_{k,j} \leftarrow x_{k,j}/r_{kk}$;
    **for** $i \in [1:k-1]$, $j \in \mathcal{M}$ **do** $x_{i,j} \leftarrow x_{i,j} - r_{ik}x_{k,j}$
  **end**
```

Figure 5.11: Solving triangular systems with a dense matrix.

The first can be solved directly, the second by applying the procedure recursively to the submatrix starting in the second row and column of L. Using an index $k \in [1:n]$ to keep track of the current submatrix, we obtain the algorithm given in Figure 5.11. Similar to `addeval_hmatrix`, it works with multiple right-hand sides represented by matrices $X \in \mathbb{K}^{\hat{t}\times\mathcal{M}}$ and overwrites the right-hand side by the solution.

Lemma 5.28 (Complexity) *The algorithm `lowersolve_amatrix` takes $|\hat{t}|^2|\mathcal{M}|$ operations to solve $LX = Y$.*

The algorithm `uppersolve_amatrix` takes $|\hat{t}|^2|\mathcal{M}|$ operations to solve $RX = Y$.

Proof. Let $j \in \mathcal{M}$. For the j-th column, the algorithm `lowersolve_amatrix` requires

$$\sum_{k=1}^{n} 1 + 2(n-k) = n + \sum_{m=0}^{n-1} m = n + 2\frac{n(n-1)}{2}$$
$$= n(n-1) + n = n^2 \text{ operations.}$$

By the same argument, we find that the algorithm `uppersolve_amatrix` requires

$$\sum_{k=1}^{n} 1 + 2(k-1) = n + 2\sum_{m=0}^{n-1} m = n + n(n-1) = n^2 \text{ operations.}$$

Multiplying by the number of columns yields the result. ∎

Case 2: t is not a leaf. In this case (t,t) has to be subdivided, i.e., we have $\operatorname{chil}(t,t) = \operatorname{chil}(t) \times \operatorname{chil}(t)$. As in the case of the inversion algorithm, we want to denote the children of t by $\{t_1, \dots, t_n\}$ with $n = |\operatorname{chil}(t)|$, but we have to preserve the triangular structure.

Lemma 5.29 (Ordered children) *Let $t \in \mathcal{T}_\mathcal{I}$ and $n := |\operatorname{chil}(t)|$. There are $t_1, \dots, t_n \in \mathcal{T}_\mathcal{I}$ such that $\operatorname{chil}(t) = \{t_1, \dots, t_n\}$ and*

$$\nu < \mu \Longrightarrow \forall i \in \hat{t}_\nu,\ j \in \hat{t}_\mu \ :\ i < j \qquad \text{for all } \nu, \mu \in [1:n].$$

Proof. We prove for all $m \in \mathbb{N}_0$ that for all $\sigma \subseteq \operatorname{chil}(t)$ with $|\sigma| = m$ we can find $t_1, \ldots, t_m \in \sigma$ such that $\sigma = \{t_1, \ldots, t_m\}$ and

$$\nu < \mu \Longrightarrow \forall i \in \hat{t}_\nu,\ j \in \hat{t}_\mu \ :\ i < j \qquad \text{for all } \nu, \mu \in [1:m] \tag{5.16}$$

by applying induction to m.

For $\sigma \subseteq \operatorname{chil}(t)$ with $|\sigma| = 0$, i.e., $\sigma = \emptyset$, the statement is trivial.

Let now $m \in \mathbb{N}_0$ be such that our claim holds for all $\sigma \subseteq \operatorname{chil}(t)$ with $|\sigma| = m$.

Let $\sigma \subseteq \operatorname{chil}(t)$ with $|\sigma| = m + 1$. Since $\mathcal{I}$ is a totally ordered set, we can find

$$j_{m+1} := \max\{j \in \hat{t}' \ :\ t' \in \sigma\},$$

and there exists $t_{m+1} \in \sigma$ with $j_{m+1} \in \hat{t}_{m+1}$. We let $\sigma' := \sigma \setminus \{t_{m+1}\}$ and observe $|\sigma'| = |\sigma| - 1 = m$, so we can apply the induction assumption to find $t_1, \ldots, t_m \in \sigma'$ satisfying (5.16).

By construction, we have $\{t_1, \ldots, t_m, t_{m+1}\} = \sigma$. Let $\nu, \mu \in [1 : m+1]$ with $\nu < \mu$. If $\nu, \mu \leq m$, (5.16) yields $i < j$ for all $i \in \hat{t}_\nu$ and $j \in \hat{t}_\mu$. Otherwise, we have $\nu < \mu = m+1$. Due to the Definition 3.14 of the cluster tree, $j_{m+1} \in \hat{t}_{m+1}$ implies $j_{m+1} \not\in \hat{t}_\nu$, and since j_{m+1} was chosen as a maximum, we have $i < j_{m+1}$ for all $i \in \hat{t}_\nu$. Definition 5.26 yields $i < j$ for all $i \in \hat{t}_\nu$ and all $j \in \hat{t}_\mu = t_{m+1}$. ■

Let $t_1, \ldots, t_n \in \operatorname{chil}(t)$ be as in Lemma 5.29. We define

$$L_{\nu\mu} := L|_{\hat{t}_\nu \times \hat{t}_\mu}, \qquad X_\mu := X|_{\hat{t}_\mu}, \qquad Y_\nu := Y|_{\hat{t}_\nu} \qquad \text{for all } \nu, \mu \in [1:n]$$

and observe

$$\nu < \mu \Longrightarrow L_{\nu\mu} = 0 \qquad \text{for all } \nu, \mu \in [1:n],$$

since L is left lower triangular, so we have

$$L|_{\hat{t}\times\hat{t}} = \begin{pmatrix} L_{11} & & \\ \vdots & \ddots & \\ L_{n1} & \ldots & L_{nn} \end{pmatrix}, \qquad X = \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix}, \qquad X = \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix}.$$

As in the case of the leaf cluster, we let

$$L_{*1} := \begin{pmatrix} L_{21} \\ \vdots \\ L_{n1} \end{pmatrix}, \quad L_{**} := \begin{pmatrix} L_{22} & & \\ \vdots & \ddots & \\ L_{n2} & \ldots & L_{nn} \end{pmatrix}, \quad X_* := \begin{pmatrix} X_2 \\ \vdots \\ X_n \end{pmatrix}, \quad Y_* := \begin{pmatrix} Y_2 \\ \vdots \\ Y_n \end{pmatrix}$$

and find

$$\begin{pmatrix} L_{11} & \\ L_{*1} & L_{**} \end{pmatrix} \begin{pmatrix} X_1 \\ X_* \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_* \end{pmatrix},$$

which is equivalent to

$$L_{11} X_1 = Y_1, \qquad L_{**} X_* = Y_* - L_{*1} X_1.$$

```
procedure lowersolve_hmatrix($L$, $t$, var $X$);
  if chil($t$) $= \emptyset$ then
    lowersolve_amatrix($N_{(t,t)}$, $X$);
  else
    for $k = 1$ to $n$ do begin
      lowersolve_hmatrix($L$, $t_k$, $X$);
      for $i \in [k+1 : n]$ do addeval_hmatrix($-1$, $L$, $(t_i, t_k)$, $X$, $X$)
    end

procedure uppersolve_hmatrix($R$, $t$, var $X$);
  if chil($t$) $= \emptyset$ then
    uppersolve_amatrix($N_{(t,t)}$, $X$);
  else
    for $k = n$ downto $1$ do begin
      uppersolve_hmatrix($R$, $t_k$, $X$);
      for $i \in [1 : k-1]$ do addeval_hmatrix($-1$, $R$, $(t_i, t_k)$, $X$, $X$)
    end
```

Figure 5.12: Solving triangular systems with a hierarchical matrix.

The first equation can be solved by applying the procedure recursively to $L_{11} = L|_{\hat{t}_1 \times t_1}$, the second equation by applying it recursively to the submatrix L_{**}. The second step leads to submatrices of the form

$$\begin{pmatrix} L_{kk} & & \\ \vdots & \ddots & \\ L_{nk} & \dots & L_{nn} \end{pmatrix},$$

so a single index $k \in [1 : n]$ is sufficient to keep track ot the submatrix, just as in the case of the inversion algorithm. The resulting algorithm is summarized in Figure 5.12.

Lemma 5.30 (Complexity) *Let $t \in \mathcal{T}_{\mathcal{I}}$. The number of operations required by the function* ***lowersolve_hmatrix*** *to solve $L|_{\hat{t} \times \hat{t}} X = Y$ is bounded by*

$$W_{lsv}(t, \mathcal{M}) := \begin{cases} |\hat{t}|^2 |\mathcal{M}| & \textit{if } \operatorname{chil}(t) = \emptyset, \\ \sum_{t' \in \operatorname{chil}(t)} W_{lsv}(t', \mathcal{M}) + \sum_{\substack{t', s' \in \operatorname{chil}(t) \\ t' > s'}} W_{mv}(t', s', \mathcal{M}) & \textit{otherwise.} \end{cases}$$

The number of operations required by the function ***uppersolve_hmatrix*** *to solve $R|_{\hat{t} \times \hat{t}} x = y$ is bounded by*

$$W_{rsv}(t, \mathcal{M}) := \begin{cases} |\hat{t}|^2 |\mathcal{M}| & \textit{if } \operatorname{chil}(t) = \emptyset, \\ \sum_{t' \in \operatorname{chil}(t)} W_{rsv}(t', \mathcal{M}) + \sum_{\substack{t', s' \in \operatorname{chil}(t) \\ t' < s'}} W_{mv}(t', s', \mathcal{M}) & \textit{otherwise.} \end{cases}$$

We have

$$W_{lsv}(t,\mathcal{M}) + W_{rsv}(t,\mathcal{M}) = W_{mv}(t,t,\mathcal{M}).$$

Proof. We prove the first claim by induction on $|\mathcal{T}_t|$.

If $|\mathcal{T}_t| = 1$, we have $\operatorname{chil}(t) = \emptyset$ and the bounds $W_{\text{lsv}}(t,\mathcal{M})$ and $W_{\text{rsv}}(t,\mathcal{M})$ have already been obtained in Lemma 5.28.

Let now $n \in \mathbb{N}$ be given such that $W_{\text{lsv}}(t,\mathcal{M})$ and $W_{\text{rsv}}(t,\mathcal{M})$ are bounds for the computational work of `lowersolve_hmatrix` and `uppersolve_hmatrix` for all $t \in \mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t| \leq n$.

Let $t \in \mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t| = n+1$. This implies $\operatorname{chil}(t) \neq \emptyset$. Since $|\mathcal{T}_{t'}| \leq n$ holds for all $t' \in \operatorname{chil}(t)$, we can use the induction assumption to find that the number of operations for `lowersolve_hmatrix` is bounded by

$$\sum_{k=1}^{n} W_{\text{lsv}}(t_k,\mathcal{M}) + \sum_{i=k+1}^{n} W_{\text{mv}}(t_i,t_k,\mathcal{M}) = \sum_{t'\in\operatorname{chil}(t)} W_{\text{lsv}}(t',\mathcal{M}) + \sum_{\substack{t',s'\in\operatorname{chil}(t)\\ t'>s'}} W_{\text{mv}}(t',s',\mathcal{M}),$$

and the number of operations for `uppersolve_hmatrix` by

$$\sum_{k=1}^{n} W_{\text{rsv}}(t_k,\mathcal{M}) + \sum_{i=1}^{k-1} W_{\text{mv}}(t_i,t_k,\mathcal{M}) = \sum_{t'\in\operatorname{chil}(t)} W_{\text{rsv}}(t',\mathcal{M}) + \sum_{\substack{t',s'\in\operatorname{chil}(t)\\ t'<s'}} W_{\text{mv}}(t',s',\mathcal{M}).$$

For the second claim, we also use induction on $|\mathcal{T}_t|$.

If $|\mathcal{T}_t| = 1$, we have $\operatorname{chil}(t) = \emptyset$ and find

$$W_{\text{lsv}}(t,\mathcal{M}) + W_{\text{rsv}}(t,\mathcal{M}) = |\hat{t}|^2|\mathcal{M}| + |\hat{t}|^2|\mathcal{M}| = 2\,|\hat{t}|^2|\mathcal{M}| = W_{\text{mv}}(t,t,\mathcal{M}).$$

Let now $n \in \mathbb{N}$ be given such that $W_{\text{lsv}}(t,\mathcal{M}) + W_{\text{rsv}}(t,\mathcal{M}) = W_{\text{mv}}(t,t)$ holds for all $t \in \mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t| \leq n$.

Let $t \in \mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t| = n+1$. This implies $\operatorname{chil}(t) \neq \emptyset$ and we have

$$W_{\text{lsv}}(t,\mathcal{M}) + W_{\text{rsv}}(t,\mathcal{M}) = \sum_{t'\in\operatorname{chil}(t)} (W_{\text{lsv}}(t',\mathcal{M}) + W_{\text{rsv}}(t',\mathcal{M})) + \sum_{\substack{t',s'\in\operatorname{chil}(t)\\ t'\neq s'}} W_{\text{mv}}(t',s',\mathcal{M}),$$

and due to $|\mathcal{T}_{t'}| \leq n$, we can use the induction assumption to find $W_{\text{lsv}}(t',\mathcal{M}) + W_{\text{rsv}}(t',\mathcal{M}) = W_{\text{mv}}(t',t',\mathcal{M})$ for all $t' \in \operatorname{chil}(t)$ and conclude

$$\begin{aligned} W_{\text{lsv}}(t,\mathcal{M}) + W_{\text{rsv}}(t,\mathcal{M}) &= \sum_{t'\in\operatorname{chil}(t)} W_{\text{mv}}(t',t',\mathcal{M}) + \sum_{\substack{t',s'\in\operatorname{chil}(t)\\ t'\neq s'}} W_{\text{mv}}(t',s',\mathcal{M}) \\ &= \sum_{t',s'\in\operatorname{chil}(t)} W_{\text{mv}}(t',s',\mathcal{M}) = W_{\text{mv}}(t,t,\mathcal{M}). \end{aligned}$$

∎

Corollary 5.31 (Complexity) *Let $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ be admissible and C_{sp}-sparse. Then calling* `lowersolve_hmatrix` *and* `uppersolve_hmatrix` *to solve $GX = LRX = B$ does not require more than*

$$4C_{\text{sp}} \max\{k, r_{\mathcal{I}}\}(p_{\mathcal{I}} + 1)|\mathcal{I}|\,|\mathcal{M}| \text{ operations.}$$

Proof. Combine Lemma 5.30 with Theorem 5.6 and bound the depth of $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ by $p_{\mathcal{I}}$. ∎

Solving block systems

Now that we have efficient algorithms for solving triangular systems at our disposal, we can consider the construction of appropriate factorizations. As an intermediate step, we require an algorithm that solves

$$LX = Y$$

with a left lower triangular $\mathcal{H}$-matrix L and an $\mathcal{H}$-matrix Y, where we are looking for an $\mathcal{H}$-matrix approximation of X. In order to be able to use recursion once more, we consider the subproblem

$$L|_{\hat{t}\times\hat{t}} X|_{\hat{t}\times\hat{s}} = Y|_{\hat{t}\times\hat{s}} \tag{5.17}$$

for $(t,t) \in \mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ and $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$.

Case 1: $b = (t,s)$ **is a leaf.** If b is an inadmissible leaf, we have $Y|_{\hat{t}\times\hat{s}} = N_{Y,b}$ and $X|_{\hat{t}\times\hat{s}} = N_{X,b}$ and can apply `lowersolve_hmatrix` to $N_{Y,b}$ to find $N_{X,b}$.

If b is an admissible leaf, we have $Y|_{\hat{t}\times\hat{s}} = A_{Y,b}B_{Y,b}^*$ and can apply `lowersolve_hmatrix` to $A_{Y,b}$ to find $A_{X,b}$ with

$$L|_{\hat{t}\times\hat{t}} A_{X,(t,s)} = A_{Y,(t,s)},$$

and using $B_{X,b} := B_{Y,b}$ and $X|_{\hat{t}\times\hat{s}} := A_{X,b}B_{X,b}^*$ yields

$$L|_{\hat{t}\times\hat{t}} X|_{\hat{t}\times\hat{s}} = L|_{\hat{t}\times\hat{t}} A_{X,b}B_{X,b}^* = A_{Y,b}B_{X,b}^* = A_{Y,b}B_{Y,b}^* = Y|_{\hat{t}\times\hat{s}}.$$

Case 2: $b = (t,s)$ **is not a leaf.** We let $n := |\operatorname{chil}(t)|$ and use Lemma 5.29 again to find $t_1, \ldots, t_n \in \operatorname{chil}(t)$ with $\operatorname{chil}(t) = \{t_1, \ldots, t_n\}$ and

$$\nu < \mu \Longrightarrow \forall i \in \hat{t}_\nu,\ j \in \hat{t}_\mu \ :\ i < j \qquad\qquad \text{for all } \nu, \mu \in [1:n].$$

We let $m := |\operatorname{chil}(s)|$ and $\operatorname{chil}(s) = \{s_1, \ldots, s_m\}$ and define

$$L_{\nu\mu} := L|_{\hat{t}_\nu\times\hat{t}_\mu},\quad X_{\mu\kappa} := X|_{\hat{t}_\nu\times\hat{s}_\kappa},\quad Y_{\mu\kappa} := Y|_{\hat{t}_\nu\times\hat{s}_\kappa} \quad \text{for all } \nu,\mu \in [1:n],\ \kappa \in [1:m]$$

and obtain

$$L|_{\hat{t}\times\hat{t}} = \begin{pmatrix} L_{11} & & \\ \vdots & \ddots & \\ L_{n1} & \ldots & L_{nn} \end{pmatrix}$$

```
procedure lsolve_hmatrix($L$, $b = (t, s)$, var $X$);
  if chil($b$) $= \emptyset$ then begin
    if $b$ admissible then
      lowersolve_hmatrix($L$, $t$, $A_{X,b}$)
    else
      lowersolve_hmatrix($L$, $t$, $N_{X,b}$)
  end else
    for $k = 1$ to $n$ do
      for $s' \in$ chil($s$) do begin
        lsolve_hmatrix($L$, $(t_k, s')$, $X$);
        for $i \in [k+1 : n]$ do addmul_hmatrix($t_i$, $t_k$, $s'$, $-1$, $L$, $X$, $X$)
      end
```

Figure 5.13: Solve $L|_{\hat{t}\times\hat{t}} X|_{\hat{t}\times\hat{s}} = Y|_{\hat{t}\times\hat{s}}$.

as before and

$$X|_{\hat{t}\times\hat{s}} = \begin{pmatrix} X_{11} & \dots & X_{1m} \\ \vdots & \ddots & \vdots \\ X_{n1} & \dots & X_{nm} \end{pmatrix}, \qquad Y|_{\hat{t}\times\hat{s}} = \begin{pmatrix} Y_{11} & \dots & Y_{1m} \\ \vdots & \ddots & \vdots \\ Y_{n1} & \dots & Y_{nm} \end{pmatrix}.$$

We can solve the system

$$\begin{pmatrix} L_{11} & & \\ \vdots & \ddots & \\ L_{n1} & \dots & L_{nn} \end{pmatrix} \begin{pmatrix} X_{11} & \dots & X_{1m} \\ \vdots & \ddots & \vdots \\ X_{n1} & \dots & X_{nm} \end{pmatrix} = LX = Y = \begin{pmatrix} Y_{11} & \dots & Y_{1m} \\ \vdots & \ddots & \vdots \\ Y_{n1} & \dots & Y_{nm} \end{pmatrix}$$

by applying block forward substitution to the columns of Y and X. In order to obtain an $\mathcal{H}$-matrix approximation X, we have to replace the exact matrix products by the approximations provided by `addmul_hmatrix` and arrive at the algorithm `lsolve_hmatrix` given in Figure 5.13.

We also require an algorithm that solves

$$XR = Y$$

with a right upper triangular $\mathcal{H}$-matrix R and an $\mathcal{H}$-matrix Y, where we are looking for an $\mathcal{H}$-matrix approximation of X. Since we will again use recursion, we consider the subproblem

$$X|_{\hat{t}\times\hat{s}} R|_{\hat{s}\times\hat{s}} = Y|_{\hat{t}\times\hat{s}} \tag{5.18}$$

for $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ and $(s, s) \in \mathcal{T}_{\mathcal{J}\times\mathcal{J}}$.

Case 1: $b = (t, s)$ is a leaf. If b is an inadmissible leaf, we have $Y|_{\hat{t}\times\hat{s}} = N_{Y,b}$ and $X|_{\hat{t}\times\hat{s}} = N_{X,b}$ and are interested in solving

$$N_{X,b} R|_{\hat{t}\times\hat{t}} = N_{Y,b} \quad \iff \quad R|^{*}_{\hat{t}\times\hat{t}} N^{*}_{X,b} = N^{*}_{Y,b}.$$

```
procedure lowersolvetrans_amatrix($R$, var $X$);
  for $k = 1$ to $n$ do begin
    for $j \in \mathcal{M}$ do $x_{k,j} \leftarrow x_{k,j}/\bar{r}_{kk}$;
    for $i \in [k+1:n]$, $j \in \mathcal{M}$ do $x_{i,j} \leftarrow x_{i,j} - \bar{r}_{ki} x_{k,j}$
  end

procedure lowersolvetrans_hmatrix($R$, $t$, var $X$);
  if $\mathrm{chil}(t) = \emptyset$ then
    lowersolvetrans_amatrix($N_{(t,t)}$, $X$);
  else
    for $k = 1$ to $n$ do begin
      lowersolvetrans_hmatrix($R$, $t_k$, $X$);
      for $i \in [k+1:n]$ do addevaltrans_hmatrix($-1$, $R$, $(t_k, t_i)$, $X$, $X$)
    end
```

Figure 5.14: Solve adjoint systems $R|_{\hat{t}\times\hat{t}}^* X = Y$ with a right upper triangular hierarchical matrix R.

Since $R|_{\hat{t}\times\hat{t}}^*$ is again a left lower triangular matrix, we can use the same approach as in `lowersolve_hmatrix`, only for the adjoint matrix $R|_{\hat{t}\times\hat{t}}^*$ instead of $L|_{\hat{t}\times\hat{t}}$. The corresponding algorithms are summarized in Figure 5.14, and applying them to the adjoint matrix $N_{Y,b}^*$ yields $N_{X,b}^*$, i.e., $N_{X,b}$.

If b is an admissible leaf, we have $Y|_{\hat{t}\times\hat{s}} = A_{Y,b} B_{Y,b}^*$ and can solve

$$B_{X,b}^* R|_{\hat{t}\times\hat{t}} = B_{Y,b}^* \qquad \Longleftrightarrow \qquad R|_{\hat{t}\times\hat{t}}^* B_{X,b} = B_{Y,b}$$

using `lowersolvetrans_hmatrix` and let $X|_{\hat{t}\times\hat{s}} = A_{Y,b} B_{X,b}^*$.

Case 2: (t,s) is not a leaf. We let $n := |\mathrm{chil}(s)|$ and use Lemma 5.29 to find $s_1, \ldots, s_n \in \mathrm{chil}(s)$ such that $\mathrm{chil}(s) = \{s_1, \ldots, s_n\}$ and

$$\nu < \mu \Longrightarrow \forall i \in \hat{t}_\nu, j \in \hat{t}_\mu \ : \ i < j \qquad\qquad \text{for all } \nu, \mu \in [1:n].$$

We let $m := |\mathrm{chil}(t)|$ and $\mathrm{chil}(t) = \{t_1, \ldots, t_m\}$, define

$$R_{\nu\mu} := L|_{\hat{s}_\nu\times\hat{s}_\mu}, \quad X_{\kappa\nu} := X|_{\hat{t}_\kappa\times\hat{s}_\nu}, \quad Y_{\kappa\nu} := Y|_{\hat{t}_\kappa\times\hat{s}_\nu} \quad \text{for all } \nu, \mu \in [1:n],\ \kappa \in [1:m]$$

and obtain

$$R|_{\hat{t}\times\hat{t}} = \begin{pmatrix} R_{11} & \ldots & R_{1n} \\ & \ddots & \vdots \\ & & R_{nn} \end{pmatrix}$$

as before and

$$X|_{\hat{t}\times\hat{s}} = \begin{pmatrix} X_{11} & \ldots & X_{1n} \\ \vdots & \ddots & \vdots \\ X_{m1} & \ldots & X_{mn} \end{pmatrix}, \qquad Y|_{\hat{t}\times\hat{s}} = \begin{pmatrix} Y_{11} & \ldots & Y_{1n} \\ \vdots & \ddots & \vdots \\ Y_{m1} & \ldots & Y_{mn} \end{pmatrix}.$$

We have to solve

$$\begin{pmatrix} X_{11} & \dots & X_{1n} \\ \vdots & \ddots & \vdots \\ X_{m1} & \dots & X_{mn} \end{pmatrix} \begin{pmatrix} R_{11} & \dots & R_{1n} \\ & \ddots & \vdots \\ & & R_{nn} \end{pmatrix} = XR = Y = \begin{pmatrix} Y_{11} & \dots & Y_{1n} \\ \vdots & \ddots & \vdots \\ Y_{m1} & \dots & Y_{mn} \end{pmatrix}.$$

We handle this task again by recursion: we introduce submatrices

$$R_{1*} := \begin{pmatrix} R_{12} & \dots & R_{1n} \end{pmatrix}, \qquad R_{**} := \begin{pmatrix} R_{22} & \dots & R_{2n} \\ & \ddots & \vdots \\ & & R_{nn} \end{pmatrix},$$

$$X_{*1} := \begin{pmatrix} X_{11} \\ \vdots \\ X_{m1} \end{pmatrix}, \qquad X_{**} := \begin{pmatrix} X_{12} & \dots & X_{1n} \\ \vdots & \ddots & \vdots \\ X_{m2} & \dots & X_{mn} \end{pmatrix},$$

$$Y_{*1} := \begin{pmatrix} Y_{11} \\ \vdots \\ Y_{m1} \end{pmatrix}, \qquad Y_{**} := \begin{pmatrix} Y_{12} & \dots & Y_{1n} \\ \vdots & \ddots & \vdots \\ Y_{m2} & \dots & Y_{mn} \end{pmatrix}$$

and find

$$R = \begin{pmatrix} R_{11} & R_{1*} \\ & R_{**} \end{pmatrix}, \qquad X = \begin{pmatrix} X_{*1} & X_{**} \end{pmatrix}, \qquad Y = \begin{pmatrix} Y_{*1} & Y_{**} \end{pmatrix}.$$

Now we can write our equation in the form

$$\begin{pmatrix} X_{*1} & X_{**} \end{pmatrix} \begin{pmatrix} R_{11} & R_{1*} \\ & R_{**} \end{pmatrix} = XR = Y = \begin{pmatrix} Y_{*1} & Y_{**} \end{pmatrix},$$

which is equivalent to

$$X_{*1} R_{11} = Y_{*1}, \qquad X_{**} R_{**} = Y_{**} - X_{*1} R_{1*}.$$

Once again we have reduced the original problem to problems for submatrices and can proceed as before in order to obtain the algorithm given in Figure 5.15.

Lemma 5.32 (Complexity) *Let $(t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$. The number of operations required by the function* ***lsolve_hmatrix*** *to solve $L|_{\hat{t}\times\hat{t}} X|_{\hat{t}\times\hat{s}} = Y|_{\hat{t}\times\hat{s}}$ is bounded by*

$$W_{lso}(t,s) := \begin{cases} W_{lsv}(t,[1:k]) & \text{if } (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}, \\ W_{lsv}(t,\hat{s}) & \text{if } (t,s) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}, \\ \sum_{\substack{t'\in\operatorname{chil}(t) \\ s'\in\operatorname{chil}(s)}} W_{lso}(t',s') + \sum_{\substack{t',r'\in\operatorname{chil}(t), \\ s'\in\operatorname{chil}(s),\ r'>t'}} W_{mm}(r',t',s') & \text{otherwise.} \end{cases}$$

```
procedure rsolve_hmatrix($R$, $t$, $s$, var $X$);
  if $\mathrm{chil}(t,s)=\emptyset$ then begin
    if $(t,s)$ admissible then
      lowersolvetrans_hmatrix($R$, $s$, $B_{X,b}$)
    else
      lowersolvetrans_hmatrix($R$, $s$, $N^*_{X,b}$)
  end else
    for $k=1$ to $n$ do
      for $t'\in\mathrm{chil}(t)$ do begin
        rsolve_hmatrix($R$, $t'$, $s_k$, $X$);
        for $i\in[k+1:n]$ do addmul_hmatrix($t'$, $s_k$, $s_i$, $-1$, $X$, $R$, $X$)
      end
```

Figure 5.15: Solve $X|_{\hat t\times\hat s}R|_{\hat s\times\hat s}=Y|_{\hat t\times\hat s}$.

The number of operations required by the function **rsolve_hmatrix** *to solve* $X|_{\hat t\times\hat s}R_{\hat s\times\hat s}=Y|_{\hat t\times\hat s}$ *is bounded by*

$$W_{rso}(t,s):=\begin{cases} W_{rsv}(s,[1:k]) & \text{if } (t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},\\ W_{rsv}(s,\hat t) & \text{if } (t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}},\\ \sum_{\substack{s'\in\mathrm{chil}(t),\\ t'\in\mathrm{chil}(t)}} W_{rso}(t',s')+\sum_{\substack{s',r'\in\mathrm{chil}(s),\\ t'\in\mathrm{chil}(t),\ r'>s'}} W_{mm}(t',s',r') & \text{otherwise.}\end{cases}$$

We have

$$W_{lso}(t,s)\le W_{mm}(t,t,s),\qquad W_{rso}(t,s)\le W_{mm}(t,s,s).$$

Proof. We prove the estimate for `lsolve_hmatrix` by induction on $|\mathcal{T}_{(t,s)}|$.

If $|\mathcal{T}_{(t,s)}|=1$, we have $\mathrm{chil}(t,s)=\emptyset$, i.e., $b=(t,s)$ is a leaf. If it is an admissible leaf, the function `lsolve_hmatrix` calls `lowersolve_hmatrix` for the matrix $A_{X,b}$, and due to Lemma 5.30 and Theorem 5.6, we obtain

$$W_{\mathrm{lso}}(t,s)\le W_{\mathrm{lsv}}(t,[1:k])\le W_{\mathrm{mv}}(t,t,[1:k])\le 2\hat k^2 B_{\mathcal{I}\times\mathcal{I}}(t,t)\le W_{\mathrm{mm}}(t,t,s).$$

If b is an inadmissible leaf, the function `lsolve_hmatrix` calls `lowersolve_hmatrix` for the matrix $N_{X,b}$ and we have

$$W_{\mathrm{lso}}(t,s)\le W_{\mathrm{lsv}}(t,\hat s)\le W_{\mathrm{mv}}(t,t,\hat s).$$

Since $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is an admissible block tree, either t or s have to be leaves. If s is a leaf, we have $|\hat s|\le\hat k$ and can use Theorem 5.6 to get

$$W_{\mathrm{lso}}(t,s)\le W_{\mathrm{mv}}(t,t,\hat s)\le 2\hat k^2 B_{\mathcal{I}\times\mathcal{I}}(t,t)\le W_{\mathrm{mm}}(t,t,s).$$

Otherwise, i.e., if t is a leaf, we have $|\hat t|\le\hat k$ and

$$W_{\mathrm{mv}}(t,t,\hat s)=2\hat k|\hat s|(|\hat t|+|\hat t|)\le 4\hat k^2|\hat s|\le 2\hat k^2(B_{\mathcal{I}\times\mathcal{I}}(t,s)+B_{\mathcal{I}\times\mathcal{I}}(t,s))\le W_{\mathrm{mm}}(t,t,s).$$

Let now $n \in \mathbb{N}$ be given such that $W_{\mathrm{lso}}(t,s)$ is a bound for the computational work of `lsolve_hmatrix` for all $(t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with $|\mathcal{T}_{(t,s)}| \leq n$. Let $(t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ with $|\mathcal{T}_{(t,s)}| = n+1$. This implies $\mathrm{chil}(t,s) \neq \emptyset$. Since $|\mathcal{T}_{(t',s')}| \leq n$ holds for all $(t',s') \in \mathrm{chil}(t,s)$, the induction assumption guarantees that the number of operations for `lsolve_hmatrix` is bounded by

$$\sum_{k=1}^{n} \sum_{s'\in\mathrm{chil}(s)} W_{\mathrm{lso}}(t_k,s') + \sum_{k=1}^{n} \sum_{i=k+1}^{n} \sum_{s'\in\mathrm{chil}(s)} W_{\mathrm{mm}}(t_i,t_k,s') = W_{\mathrm{lso}}(t,s),$$

and we find

$$\begin{aligned}
W_{\mathrm{lso}}(t,s) &= \sum_{k=1}^{n} \sum_{s'\in\mathrm{chil}(s)} W_{\mathrm{lso}}(t_k,s') + \sum_{k=1}^{n} \sum_{i=k+1}^{n} \sum_{s'\in\mathrm{chil}(s)} W_{\mathrm{mm}}(t_i,t_k,s') \\
&\leq \sum_{k=1}^{n} \sum_{s'\in\mathrm{chil}(s)} W_{\mathrm{mm}}(t_k,t_k,s') + \sum_{k=1}^{n} \sum_{i=k+1}^{n} \sum_{s'\in\mathrm{chil}(s)} W_{\mathrm{mm}}(t_i,t_k,s') \\
&= \sum_{k=1}^{n} \sum_{i=k}^{n} \sum_{s'\in\mathrm{chil}(s)} W_{\mathrm{mm}}(t_i,t_k,s') \\
&\leq \sum_{i,k=1}^{n} \sum_{s'\in\mathrm{chil}(s)} W_{\mathrm{mm}}(t_i,t_k,s') \\
&= \sum_{t',r'\in\mathrm{chil}(t)} \sum_{s'\in\mathrm{chil}(s)} W_{\mathrm{mm}}(r',t',s') \leq W_{\mathrm{mm}}(t,t,s).
\end{aligned}$$

The result for `rsolve_hmatrix` can be obtained by similar arguments, taking into account that `lsolvetrans_hmatrix` works with the right upper triangular part of $R|_{\hat{t}\times\hat{t}}$, therefore the complexity for the leaf cases is bounded by $W_{\mathrm{rsv}}(t,\mathcal{M})$ instead of $W_{\mathrm{lsv}}(t,\mathcal{M})$. ■

Finding the LR factorization

Now that we know how to solve linear systems involving triangular matrices, we have to find a way to find a decomposition of a given $\mathcal{H}$-matrix into triangular factors. We focus on the LR factorization

$$LR = G,$$

bearing in mind that other triangular factorizations, e.g., the Cholesky factorization, can be constructed by a similar approach. As before, we consider a subproblem

$$L|_{\hat{t}\times\hat{t}} R|_{\hat{t}\times\hat{t}} = G|_{\hat{t}\times\hat{t}} \tag{5.19}$$

for $t \in \mathcal{T}_{\mathcal{I}}$, where $G|_{\hat{t}\times\hat{t}}$ is given and we are looking for $L|_{\hat{t}\times\hat{t}}$ and $R|_{\hat{t}\times\hat{t}}$.

Let $t \in \mathcal{T}_{\mathcal{I}}$. If $\mathrm{chil}(t) = \emptyset$ holds, we can compute the LR factorization by the well-known algorithm given in Figure 5.16, where we again enumerate the indices $\hat{t} = \{i_1,\ldots,i_n\}$ with $n = |\hat{t}|$ and let $g_{\nu\mu} = g_{i_\nu,i_\mu}$.

```
procedure lrdecomp_amatrix(var $G$);
  for $k = 1$ to $n$ do begin
    for $i \in [k+1 : n]$ do $g_{ik} \leftarrow g_{ik}/g_{kk}$;
    for $i, j \in [k+1 : n]$ do $g_{ij} \leftarrow g_{ij} - g_{ik}g_{kj}$
  end
```

Figure 5.16: Computing the LR factorization $LR = G$ of a matrix in standard array representation. The upper triangular part of G is overwritten by R, the strictly lower triangular part by L. All diagonal elements of L are equal to one and are not stored.

Let now $\operatorname{chil}(t) \neq \emptyset$. We let $n := |\operatorname{chil}(t)|$ and use Lemma 5.29 again to find $t_1, \dots, t_n \in \operatorname{chil}(t)$ such that $\operatorname{chil}(t) = \{t_1, \dots, t_n\}$ and

$$\nu < \mu \Longrightarrow \forall i \in \hat{t}_\nu, j \in \hat{t}_\mu \ : \ i < j \qquad \text{for all } \nu, \mu \in [1 : n].$$

As in the previous sections, we split G, L and R into block matrices by defining

$$G_{\nu\mu} := G|_{\hat{t}_\nu \times \hat{t}_\mu}, \qquad L_{\nu\mu} := L|_{\hat{t}_\nu \times \hat{t}_\mu}, \qquad R_{\nu\mu} := R|_{\hat{t}_\nu \times \hat{t}_\mu} \qquad \text{for all } \nu, \mu \in [1 : n]$$

and obtain

$$G|_{\hat{t}\times\hat{t}} = \begin{pmatrix} G_{11} & \dots & G_{1n} \\ \vdots & \ddots & \vdots \\ G_{n1} & \dots & G_{nn} \end{pmatrix},$$

$$L|_{\hat{t}\times\hat{t}} = \begin{pmatrix} L_{11} & & \\ \vdots & \ddots & \\ L_{n1} & \dots & L_{nn} \end{pmatrix}, \qquad R|_{\hat{t}\times\hat{t}} = \begin{pmatrix} R_{11} & \dots & R_{1n} \\ & \ddots & \vdots \\ & & R_{nn} \end{pmatrix}.$$

We aim to construct the LR factorization recursively by splitting the matrices into the first row and column and a smaller remainder. We introduce

$$G_{*1} := \begin{pmatrix} G_{21} \\ \vdots \\ G_{n1} \end{pmatrix}, \qquad G_{1*} := \begin{pmatrix} G_{12} & \dots & G_{1n} \end{pmatrix}, \qquad G_{**} := \begin{pmatrix} G_{22} & \dots & G_{2n} \\ \vdots & \ddots & \vdots \\ G_{n2} & \dots & G_{nn} \end{pmatrix},$$

$$L_{*1} := \begin{pmatrix} L_{21} \\ \vdots \\ L_{n1} \end{pmatrix}, \qquad L_{**} := \begin{pmatrix} L_{22} & & \\ \vdots & \ddots & \\ L_{n2} & \dots & L_{nn} \end{pmatrix},$$

$$R_{1*} := \begin{pmatrix} R_{12} & \dots & R_{1n} \end{pmatrix}, \qquad R_{**} := \begin{pmatrix} R_{22} & \dots & R_{2n} \\ & \ddots & \vdots \\ & & R_{nn} \end{pmatrix}$$

```
procedure lrdecomp_hmatrix($t$, var $G$);
  if chil($t$) $= \emptyset$ then
    lrdecomp_amatrix($N_{G,(t,t)}$)
  else
    for $k = 1$ to $n$ do begin
      lrdecomp_hmatrix($t_k$, $G$);
      for $j \in [k+1:n]$ do lsolve_hmatrix($G$, $t_k$, $t_j$, $G$);
      for $i \in [k+1:n]$ do rsolve_hmatrix($G$, $t_i$, $t_k$, $G$);
      for $i, j \in [k+1:n]$ do addmul_hmatrix($t_i$, $t_k$, $t_j$, $-1$, $G$, $G$, $G$)
    end
```

Figure 5.17: Computing the approximate LR factorization $LR = G$ of an $\mathcal{H}$-matrix. The upper triangular part of G is overwritten by R, the strictly lower triangular part by L. All diagonal elements of L are equal to one and do not have to be stored.

and write the equation (5.19) in the compact form

$$\begin{pmatrix} L_{11} & \\ L_{*1} & L_{**} \end{pmatrix} \begin{pmatrix} R_{11} & R_{1*} \\ & R_{**} \end{pmatrix} = \begin{pmatrix} G_{11} & G_{1*} \\ G_{*1} & G_{**} \end{pmatrix}.$$

It is equivalent with the four equations

$$L_{11}R_{11} = G_{11}, \tag{5.20a}$$
$$L_{11}R_{1*} = G_{1*}, \tag{5.20b}$$
$$L_{*1}R_{11} = G_{*1}, \tag{5.20c}$$
$$L_{**}R_{**} = G_{**} - L_{*1}R_{1*}. \tag{5.20d}$$

The first equation (5.20a) can be solved by applying our algorithm recursively and yields L_{11} and R_{11}. The second equation (5.20b) can be handled by the function `lsolve_hmatrix` introduced in Figure 5.13 and yields R_{1*}. The third equation (5.20c) can be treated by the function `rsolve_hmatrix` given in Figure 5.15 and yields L_{*1}.

For the last equation (5.20d), we can use the multiplication function `addmul_hmatrix` given in Figure 5.8 to obtain the right-hand side and then apply our procedure recursively to submatrices of the form

$$\begin{pmatrix} G_{kk} & \dots & G_{kn} \\ \vdots & \ddots & \vdots \\ G_{nk} & \dots & G_{nn} \end{pmatrix},$$

so we can keep track of the submatrices using an index $k \in [1:n]$ and arrive at the algorithm given in Figure 5.17.

Theorem 5.33 (Complexity) *Let $t \in \mathcal{T}_{\mathcal{I}}$. The number of operations required by the function `lrdecomp_hmatrix` to find the approximate LR factorization of $G|_{\hat{t}\times\hat{t}}$ is bounded by*

$$W_{lr}(t) := \begin{cases} 2|\hat{t}|^3/3 & \text{if } \operatorname{chil}(t) = \emptyset, \\ \sum_{t'\in\operatorname{chil}(t)} W_{lr}(t') + \sum_{\substack{t',s'\in\operatorname{chil}(t)\\ s'>t'}} (W_{lso}(t',s') + W_{rso}(s',t')) & \text{otherwise} \\ \quad + \sum_{\substack{t',s',r'\in\operatorname{chil}(t)\\ s',r'>t'}} W_{mm}(s',t',r'). & \end{cases}$$

We have

$$W_{lr}(t) \le W_{mm}(t,t,t).$$

Proof. By induction on $|\mathcal{T}_t|$.

If $|\mathcal{T}_t = 1|$, we have $\operatorname{chil}(t) = \emptyset$ and the function calls `lrdecomp_amatrix` to compute the factorization of the matrix $N_{G,(t,t)}$. This takes not more than

$$2|\hat{t}|^3/3 \le 2|\hat{t}|^3 = W_{\mathrm{mm}}(t,t,t)$$

operations.

Let now $n \in \mathbb{N}$ be such that our claims hold for all $t \in \mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t| \le n$. Let $t \in \mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t| = n+1$. Then we have $\operatorname{chil}(t) \neq \emptyset$ and the function calls itself recursively for all $t' \in \operatorname{chil}(t)$. For each $t' \in \operatorname{chil}(t)$, we have $|\mathcal{T}_{t'}| \le n$ and can apply the induction assumption to find that the recursive call to `lrdecomp_hmatrix` requires not more than $W_{\mathrm{lr}}(t')$ operations. Combining this estimate with the estimates provided by Lemma 5.32 and Theorem 5.18 yields that the total number of operations is bounded by

$$\sum_{k=1}^{n} W_{\mathrm{lr}}(t_k) + \sum_{k=1}^{n}\sum_{i=k+1}^{n} (W_{\mathrm{lso}}(t_k,t_i) + W_{\mathrm{rso}}(t_i,t_k)) + \sum_{k=1}^{n}\sum_{i,j=k+1}^{n} W_{\mathrm{mm}}(t_i,t_k,t_j) = W_{\mathrm{lr}}(t).$$

The induction assumption also yields $W_{\mathrm{lr}}(t') \le W_{\mathrm{mm}}(t',t',t')$, and with Lemma 5.32 and Theorem 5.18 we find

$$\begin{aligned} W_{\mathrm{lr}}(t) &= \sum_{k=1}^{n} W_{\mathrm{lr}}(t_k) + \sum_{k=1}^{n}\sum_{i=k+1}^{n} (W_{\mathrm{lso}}(t_k,t_i) + W_{\mathrm{rso}}(t_i,t_k)) + \sum_{k=1}^{n}\sum_{i,j=k+1}^{n} W_{\mathrm{mm}}(t_i,t_k,t_j) \\ &\le \sum_{k=1}^{n} W_{\mathrm{mm}}(t_k,t_k,t_k) + \sum_{k=1}^{n}\sum_{i=k+1}^{n} (W_{\mathrm{mm}}(t_k,t_k,t_i) + W_{\mathrm{mm}}(t_i,t_k,t_k)) \\ &\quad + \sum_{k=1}^{n}\sum_{i,j=k+1}^{n} W_{\mathrm{mm}}(t_i,t_k,t_j) \\ &\le \sum_{i,j,k=1}^{n} W_{\mathrm{mm}}(t_i,t_j,t_k) \le W_{\mathrm{mm}}(t,t,t). \end{aligned}$$

■

6 $\mathcal{H}^2$-matrices

The efficiency of $\mathcal{H}$-matrix techniques can be significantly improved if we modify the low-rank properties used to obtain approximations: instead of handling each admissible block on its own, we are looking for factorizations of entire collections of blocks.

This approach gives rise to $\mathcal{H}^2$-*matrices*, a variant of $\mathcal{H}$-matrices that requires only $\mathcal{O}(nk)$ units of storage instead of the $\mathcal{O}(nk \log n)$ units of storage associated with standard $\mathcal{H}$-matrices. Due to this improved asymptotic behaviour, $\mathcal{H}^2$-matrices are particularly attractive for very large matrices.

6.1 Motivation

We consider the one-dimensional model problem introduced in Chapter 2 and are looking for an improved approximation of the matrix $G \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ given by

$$g_{ij} = \int_{(i-1)/n}^{i/n} \int_{(j-1)/n}^{j/n} g(x,y)\,dy\,dx \qquad \text{for all } i,j \in \mathcal{I}.$$

We use the cluster tree $\mathcal{T}_{\mathcal{I}}$ introduced in Chapter 2: a cluster $t \in \mathcal{T}_{\mathcal{I}}$ is an interval $t = [a_t, b_t]$ such that

$$[(i-1)/n, i/n] \subseteq t \qquad \text{for all } i \in \hat{t}.$$

In order to approximate a submatrix $G|_{\hat{t}\times\hat{s}}$ corresponding to clusters $t, s \in \mathcal{T}_{\mathcal{I}}$, we consider degenerate approximations of the kernel function g. We denote the Chebyshev points in the reference interval $[-1,1]$ by

$$\hat{\xi}_\nu := \cos\left(\pi \frac{2\nu+1}{2m+2}\right) \qquad \text{for all } \nu \in [0:m]$$

and the transformed Chebyshev points in the intervals $t = [a_t, b_t]$ and $s = [a_s, b_s]$ by

$$\xi_{t,\nu} := \frac{b_t + a_t}{2} + \frac{b_t - a_t}{2}\hat{\xi}_\nu, \quad \xi_{s,\mu} := \frac{b_s + a_s}{2} + \frac{b_s - a_s}{2}\hat{\xi}_\mu \qquad \text{for all } \nu, \mu \in [0:m].$$

The corresponding Lagrange polynomials are given by

$$\ell_{t,\nu}(x) := \prod_{\substack{\lambda=0\\ \lambda\neq\nu}} \frac{x - \xi_{t,\lambda}}{\xi_{t,\nu} - \xi_{t,\lambda}}, \quad \ell_{s,\mu}(y) := \prod_{\substack{\kappa=0\\ \kappa\neq\mu}} \frac{y - \xi_{s,\kappa}}{\xi_{s,\mu} - \xi_{s,\kappa}} \qquad \text{for all } \nu,\mu \in [0:m],\ x,y \in \mathbb{R}.$$

We have constructed $\mathcal{H}$-matrix approximations of $G|_{\hat{t}\times\hat{s}}$ by applying interpolation to the variable x, i.e., by using

$$\tilde{g}_{t,s}(x,y) := \sum_{\nu=0}^{m} \ell_{t,\nu}(x) g(\xi_{t,\nu}, y) \qquad \text{for all } x \in t,\ y \in s.$$

Since there is no reason to give x preferential treatment, we could also have used interpolation in the variable y, i.e.,

$$\tilde{g}_{t,s}(x,y) := \sum_{\mu=0}^{m} g(x, \xi_{s,\mu}) \ell_{s,\mu}(y) \qquad \text{for all } x \in t,\ y \in s,$$

and this would also give rise to a low-rank approximation.

The key to $\mathcal{H}^2$-matrices is to apply interpolation to *both* variables simultaneously, i.e., to use

$$\tilde{g}_{t,s}(x,y) := \sum_{\nu=0}^{m}\sum_{\mu=0}^{m} \ell_{t,\nu}(x) g(\xi_{t,\nu}, \xi_{s,\mu}) \ell_{s,\mu}(y) \qquad \text{for all } x \in t,\ y \in s.$$

This corresponds to two-dimensional tensor interpolation, and Theorem 3.6 states that the resulting error is only slighly larger than for the simple interpolation.

Discretizing this approximation leads to

$$\begin{aligned}
\tilde{g}_{ij} &:= \int_{(i-1)/n}^{i/n} \int_{(j-1)/n}^{j/n} \tilde{g}_{t,s}(x,y)\,dy\,dx \\
&= \sum_{\nu=0}^{m}\sum_{\mu=0}^{m} \int_{(i-1)/n}^{i/n} \ell_{t,\nu}(x)\,dx\ g(\xi_{t,\nu}, \xi_{s,\mu}) \int_{(j-1)/n}^{j/n} \ell_{s,\mu}(y)\,dy \\
&= \sum_{\nu=0}^{m}\sum_{\mu=0}^{m} v_{t,i\nu} s_{b,\nu\mu} w_{s,j\mu} = (V_t S_b W_s^*)_{ij} \qquad \text{for all } i \in \hat{t},\ j \in \hat{s},
\end{aligned}$$

where we introduce $V_t \in \mathbb{R}^{\hat{t}\times M}$, $W_s \in \mathbb{R}^{\hat{s}\times M}$ and $S_b \in \mathbb{R}^{M\times M}$ with $M := [0:m]$ and

$$\begin{aligned}
v_{t,i\nu} &:= \int_{(i-1)/n}^{i/n} \ell_{t,\nu}(x)\,dx, \qquad w_{s,j\mu} := \int_{(j-1)/n}^{j/n} \ell_{s,\mu}(y)\,dy, \\
s_{b,\nu\mu} &:= g(\xi_{t,\nu}, \xi_{s,\mu}) \qquad \text{for all } i \in \hat{t},\ j \in \hat{s},\ \nu,\mu \in M.
\end{aligned}$$

We observe that V_t depends only on the row cluster t, but not on the column cluster s, while W_s depends only on the column cluster s, but not on t. The entire interaction between both clusters is expressed by the *coupling matrix* S_b, and this matrix is typically small, since it has only $k := m + 1$ rows and columns.

It is of particular importance that the size of the matrices S_b does not depend on the level of the block: for $\mathcal{H}$-matrices, admissible blocks are more expensive the closer they are to the root, while for $\mathcal{H}^2$-matrices the storage requirements are constant. Since the number of blocks grows exponentially with the depth of the cluster tree, this property allows us to obtain *linear* complexity with regard to the number of indices.

Remark 6.1 (Block storage) *If we have $n = 2^q$ and choose the depth $p \in [0 : q]$ of the cluster tree such that $k < 2^{q-p} \leq 2k$, Lemma 2.7 yields that storing S_b for all admissible blocks requires*

$$\sum_{\ell=0}^{p} k^2 \, |\mathcal{A}_\ell| = 6k^2 \sum_{\ell=1}^{p} (2^{\ell-1} - 1) = 6k^2(2^p - 1 - p) < 6k2^{q-p}2^p = 6k2^q = 6kn$$

units of storage. We already know that storing N_b for all inadmissible blocks requires

$$(2^{q-p})^2(|\mathcal{D}_p| + |\mathcal{L}_p| + |\mathcal{R}_p|) \leq 2k2^{q-p}(2^p + 2^p - 1 + 2^p - 1) \leq 6k2^{q-p}2^p = 6kn$$

units of storage. This gives us a total of $12kn$ units of storage for all coupling and nearfield matrices: the storage requirements grow linearly *with n.*

In comparison, the $\mathcal{H}$-matrix approximation constructed in Chapter 2 requires more than $6k(p-2)n$ units of storage and therefore is more expensive for $p > 4$.

Unfortunately, the storage requirements for the matrices V_t and W_s do not grow linearly with n: storing V_t for a cluster t on level ℓ requires $2^{q-\ell}k$ units of storage, and since there are 2^ℓ clusters on this level, we obtain a total of

$$\sum_{\ell=0}^{p} \sum_{\substack{t \in \mathcal{T}_\mathcal{I} \\ \text{level}(t)=\ell}} 2^{q-\ell}k = \sum_{\ell=0}^{p} 2^\ell 2^{q-\ell}k = \sum_{\ell=0}^{p} 2^q k = nk(p+1).$$

In order to obtain linear growth of the storage requirements, we have to handle the families $(V_t)_{t \in \mathcal{T}_\mathcal{I}}$ and $(W_s)_{s \in \mathcal{T}_\mathcal{I}}$ of matrices more efficiently.

The coefficients of V_t are closely connected to the Lagrange polynomials $\ell_{t,\nu}$, and since our interpolation operators have the projection property (2.26), we find

$$\ell_{t,\nu} = \sum_{\nu'=0}^{m} \ell_{t,\nu}(\xi_{t',\nu'})\ell_{t',\nu'} \qquad \text{for all } t' \in \mathcal{T}_\mathcal{I}, \tag{6.1}$$

i.e., we can represent any Lagrange polynomial of a given cluster in the Lagrange basis corresponding to any other cluster.

We can apply this approach to the children of t: if $\text{chil}(t) \neq \emptyset$, Definition 3.14 yields that for any $i \in \hat{t}$ there is exactly one child $t' \in \text{chil}(t)$ such that $i \in \hat{t}'$. Applying (6.1) to this child cluster gives us

$$v_{t,i\nu} = \int_{(i-1)/n}^{i/n} \ell_{t,\nu}(x)\,dx = \sum_{\nu'=0}^{m} \ell_{t,\nu}(\xi_{t',\nu'}) \int_{(i-1)/n}^{i/n} \ell_{t',\nu'}(x)\,dx = \sum_{\nu'=0}^{m} \ell_{t,\nu}(\xi_{t',\nu'}) v_{t',i\nu'},$$

i.e., we can avoid storing $v_{t,i\nu}$ and store only the k^2 *transfer coefficients* $\ell_{t\nu}(\xi_{t',\nu'})$ instead. Since this works for all $i \in \hat{t}$, the entire matrix V_t can be expressed implicitly via the transfer coefficients.

This observation suggests the following approach: the matrices V_t are only stored explicitly for leaf clusters and expressed implicitly via *transfer matrices* $E_{t'} \in \mathbb{R}^{M\times M}$ in the form

$$V_t|_{\hat{t}'} = V_{t'}E_{t'} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}},\ t' \in \operatorname{chil}(t), \tag{6.2}$$

where we recall the notation (5.1) and the transfer coefficients are given by

$$e_{t',\nu'\nu} := \ell_{t,\nu}(\xi_{t',\nu'}) \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}},\ t' \in \operatorname{chil}(t),\ \nu,\nu' \in M.$$

In the case of the one-dimensional model problem, we have denoted the children of a non-leaf cluster t by t_1 and t_2 and can write (6.2) in the short form

$$V_t = \begin{pmatrix} V_{t_1}E_{t_1} \\ V_{t_2}E_{t_2} \end{pmatrix} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}} \text{ with } \operatorname{chil}(t) \neq \emptyset.$$

Remark 6.2 (Cluster storage) *Following this approach, storing the matrices V_t only for the leaf clusters on level p takes*

$$\sum_{\substack{t\in\mathcal{T}_{\mathcal{I}}\\ \operatorname{level}(t)=p}} 2^{q-p}k = 2^p 2^{q-p}k = nk$$

units of storage, while storing the transfer matrices E_t for all cluster takes

$$\sum_{t\in\mathcal{T}_{\mathcal{I}}} k^2 = \sum_{\ell=0}^{p} \sum_{\substack{t\in\mathcal{T}_{\mathcal{I}}\\ \operatorname{level}(t)=\ell}} k^2 = \sum_{\ell=0}^{p} 2^\ell k^2 = (2^{p+1}-1)k^2 < 2^{p+1}2^{q-p}k = 2^{q+1}k = 2nk$$

units of storage, where we have again assumed $k < 2^{q-p}$.

We conclude that transfer matrices allow us to represent the entire family $(V_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ using less than $3nk$ coefficients. The same holds for $(W_s)_{s\in\mathcal{T}_{\mathcal{I}}}$.

Since $(V_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ and $(W_s)_{s\in\mathcal{T}_{\mathcal{I}}}$ are identical, we only have to store $(V_t)_{t\in\mathcal{T}_{\mathcal{I}}}$. The coupling matrices S_b require not more than $6nk$ units of storage, the nearfield matrices N_b require also not more than $6nk$ units, and the cluster matrices V_t require not more than $3nk$ units, so the total storage requirements are bounded by $15nk$. We have reached our goal: the bound grows only linearly with n.

The reduction of the storage requirements comes at a price: given an admissible block $b = (t,s)$, we cannot directly evaluate $G|_{\hat{t}\times\hat{s}} \approx V_t S_b W_s^*$, since V_t and W_s are only at our disposal for leaf clusters, but not for any other clusters. Fortunately, we can modify the corresponding algorithms in a way that not only solves this problem but even reduces the number of operations, i.e., we not only save storage, but also time.

6.2 $\mathcal{H}^2$-matrices

As in the case of $\mathcal{H}$-matrices, we can generalize the results obtained for the one-dimensional model problem.

Let $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$ be cluster trees for index sets $\mathcal{I}$ and $\mathcal{J}$, and let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be an admissible block tree for $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$.

Definition 6.3 (Cluster basis) *Let $k \in \mathbb{N}_0$. A family $V = (V_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ of matrices is called a* cluster basis *of rank k if*

- *for all $t \in \mathcal{T}_{\mathcal{I}}$ we have $V_t \in \mathbb{K}^{\hat{t}\times k}$, and*
- *for all $t \in \mathcal{T}_{\mathcal{I}}$ and all $t' \in \operatorname{chil}(t)$ there is a matrix $E_{t'} \in \mathbb{K}^{k\times k}$ satisfying*

$$V_t|_{\hat{t}'} = V_{t'} E_{t'}. \tag{6.3}$$

In this case, we call the matrices V_t basis matrices *and the matrices E_t* transfer matrices*.*

Definition 6.4 (Nested representation) *Let $V = (V_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ be a cluster basis of rank k, and let $(E_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ be a family of transfer matrices satisfying (6.3).*

Then we call $((V_t)_{t\in\mathcal{L}_{\mathcal{I}}}, (E_t)_{t\in\mathcal{T}_{\mathcal{I}}})$ a nested representation *of the cluster basis V.*

In a nested representation, the matrices V_t are only given explicitly for leaf clusters $t \in \mathcal{L}_{\mathcal{I}}$ and otherwise implicitly by (6.3).

We can see that (6.3) determines the transfer matrix $E_{t'}$ only uniquely if $V_{t'}$ is injective.

For the root cluster $t = \operatorname{root}(\mathcal{T}_{\mathcal{I}})$, the transfer matrix E_t is never used and only included in the nested representation for the sake of brevity.

Definition 6.5 ($\mathcal{H}^2$-matrix) *Let $V = (V_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ and $W = (W_s)_{s\in\mathcal{T}_{\mathcal{J}}}$ be cluster bases of rank k. A matrix $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ is called an* $\mathcal{H}^2$-matrix *for V, W, and the block tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ if for each admissible block $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ there is a matrix $S_b \in \mathbb{K}^{k\times k}$ such that*

$$G|_{\hat{t}\times\hat{s}} = V_t S_b W_s^*. \tag{6.4}$$

In this case, V is called the row cluster basis*, W is called the* column cluster basis*, and the matrices S_b are called* coupling matrices*.*

Definition 6.6 ($\mathcal{H}^2$-matrix representation) *Let $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ be an $\mathcal{H}^2$-matrix for the row cluster basis V, the column cluster basis W, and the block tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$.*

Let $((V_t)_{t\in\mathcal{L}_{\mathcal{I}}}, (E_t)_{t\in\mathcal{T}_{\mathcal{I}}})$ and $((W_s)_{s\in\mathcal{L}_{\mathcal{J}}}, (F_s)_{s\in\mathcal{T}_{\mathcal{J}}})$ be nested representations of the cluster bases V and W.

Let $(S_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ be a family of coupling matrices satisfying (6.4) and let

$$N_b := G|_{\hat{t}\times\hat{s}} \qquad \text{for all } b = (t,s) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}.$$

Then we call $((V_t)_{t\in\mathcal{L}_{\mathcal{I}}}, (E_t)_{t\in\mathcal{T}_{\mathcal{I}}}, (W_s)_{s\in\mathcal{L}_{\mathcal{J}}}, (F_s)_{s\in\mathcal{T}_{\mathcal{J}}}, (S_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}, (N_b)_{b\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}})$ an $\mathcal{H}^2$-matrix representation *of G.*

Lemma 6.7 (Storage, cluster basis) *Let $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ be a cluster basis of rank k. A nested representation of V requires not more than*

$$k|\mathcal{I}| + k^2|\mathcal{T}_\mathcal{I}| \text{ units of storage.}$$

Proof. The nested representation stores the basis matrices $V_t \in \mathbb{K}^{\hat{t}\times k}$ for all leaf clusters $t \in \mathcal{L}_\mathcal{I}$, and this requires

$$\sum_{t\in\mathcal{L}_\mathcal{I}} k|\hat{t}| = k \sum_{t\in\mathcal{L}_\mathcal{I}} |\hat{t}| = k\Big|\bigcup_{t\in\mathcal{L}_\mathcal{I}} \hat{t}\Big| = k|\mathcal{I}| \text{ units of storage,}$$

since we have proven in Corollary 3.19 that the index sets of the leaves are a disjoint partition of $\mathcal{I}$. The nested representation also stores the transfer matrices $E_{t'} \in \mathbb{K}^{k\times k}$ for all $t \in \mathcal{T}_\mathcal{I}$ and $t' \in \mathrm{chil}(t)$, and this requires

$$\sum_{t\in\mathcal{T}_\mathcal{I}} \sum_{t'\in\mathrm{chil}(t)} k^2 \le \sum_{t'\in\mathcal{T}_\mathcal{I}} k^2 = k^2|\mathcal{T}_\mathcal{I}| \text{ units of storage,}$$

since every cluster has at most one parent. ∎

Lemma 6.8 (Storage, block matrices) *Let $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ be an $\mathcal{H}^2$-matrix for a row basis V and a column basis W of rank k. Let the block tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be C_{sp}-sparse and strictly admissible (cf. Definition 3.28).*

The nearfield matrices $(N_b)_{b\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}}$ and the coupling matrices $(S_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ or an $\mathcal{H}^2$-matrix representation of G require not more than

$$C_{sp}\min\{r_\mathcal{J}|\mathcal{I}|, r_\mathcal{I}|\mathcal{J}|\} + C_{sp}k^2\min\{|\mathcal{T}_\mathcal{I}|, |\mathcal{T}_\mathcal{J}|\} \text{ units of storage.}$$

Proof. We first consider the storage for the nearfield matrices. Let $b = (t,s) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$. The matrix N_b requires $|\hat{t}|\,|\hat{s}|$ units of storage. Since $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is *strictly* admissible, we have $t \in \mathcal{L}_\mathcal{I}$ *and* $s \in \mathcal{L}_\mathcal{J}$ and obtain

$$|\hat{t}| \le r_\mathcal{I}, \qquad\qquad |\hat{s}| \le r_\mathcal{J}.$$

The storage for all nearfield matrices is therefore bounded by

$$\begin{aligned}\sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} |\hat{t}|\,|\hat{s}| &\le r_\mathcal{J} \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} |\hat{t}| \le r_\mathcal{J} \sum_{t\in\mathcal{L}_\mathcal{I}} \sum_{s\in\mathrm{row}(t)} |\hat{t}| \\ &\le C_{\mathrm{sp}} r_\mathcal{J} \sum_{t\in\mathcal{L}_\mathcal{I}} |\hat{t}| = C_{\mathrm{sp}} r_\mathcal{J}\Big|\bigcup_{t\in\mathcal{L}_\mathcal{I}} \hat{t}\Big| = C_{\mathrm{sp}} r_\mathcal{J}|\mathcal{I}|,\end{aligned}$$

where we use Corollary 3.19 for the last two equations, and, following the same arguments, also by

$$\sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} |\hat{t}|\,|\hat{s}| \le r_\mathcal{I} \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} |\hat{s}| \le C_{\mathrm{sp}} r_\mathcal{I}|\mathcal{J}|.$$

Now we investigate the storage requirements of the coupling matrices. Let $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. The matrix S_b requires k^2 units of storage, and the storage for all coupling matrices is bounded by

$$\sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} k^2 \le \sum_{t\in\mathcal{T}_\mathcal{I}} \sum_{s\in\operatorname{row}(t)} k^2 \le k^2 \sum_{t\in\mathcal{T}_\mathcal{I}} |\operatorname{row}(t)| \le C_{\mathrm{sp}} k^2 |\mathcal{T}_\mathcal{I}|,$$
$$\sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} k^2 \le \sum_{s\in\mathcal{T}_\mathcal{J}} \sum_{t\in\operatorname{col}(s)} k^2 \le k^2 \sum_{s\in\mathcal{T}_\mathcal{J}} |\operatorname{col}(s)| \le C_{\mathrm{sp}} k^2 |\mathcal{T}_\mathcal{J}|.$$

Combining these estimates yields the required upper bound. ∎

Theorem 6.9 ($\mathcal{H}^2$-matrix representation) *Let $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ be an $\mathcal{H}^2$-matrix for a row basis V and a column basis W of rank k. Let the block tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be C_{sp}-sparse and strictly admissible. An $\mathcal{H}^2$-matrix representation of G requires not more than*

$$\frac{C_{sp}+2}{2}\max\{k, r_\mathcal{I}, r_\mathcal{J}\}(|\mathcal{I}| + |\mathcal{J}|) + \frac{C_{sp}+2}{2}k^2(|\mathcal{T}_\mathcal{I}| + |\mathcal{T}_\mathcal{J}|) \text{ units of storage.}$$

Proof. Due to Lemma 6.7, the nested representations of V and W require not more than

$$k(|\mathcal{I}| + |\mathcal{J}|) + k^2(|\mathcal{T}_\mathcal{I}| + |\mathcal{T}_\mathcal{J}|) \text{ units of storage}$$

while Lemma 6.8 guarantees that the nearfield and coupling matrices require not more than

$$\begin{aligned} &C_{\mathrm{sp}} \min\{r_\mathcal{J}|\mathcal{I}|, r_\mathcal{I}|\mathcal{J}|\} + C_{\mathrm{sp}} k^2 \min\{|\mathcal{T}_\mathcal{I}|, |\mathcal{T}_\mathcal{J}|\} \\ &\quad\le \frac{C_{\mathrm{sp}}}{2}(r_\mathcal{J}|\mathcal{I}| + r_\mathcal{I}|\mathcal{J}|) + \frac{C_{\mathrm{sp}}}{2}k^2(|\mathcal{T}_\mathcal{I}| + |\mathcal{T}_\mathcal{J}|) \\ &\quad\le \frac{C_{\mathrm{sp}}}{2}\max\{r_\mathcal{I}, r_\mathcal{J}\}(|\mathcal{I}| + |\mathcal{J}|) + \frac{C_{\mathrm{sp}}}{2}k^2(|\mathcal{T}_\mathcal{I}| + |\mathcal{T}_\mathcal{J}|) \text{ units of storage.} \end{aligned}$$

Adding both estimates yields the required upper bound. ∎

Remark 6.10 (Complexity estimate) *If we assume that $|\operatorname{chil}(t)| \ne 1$ holds for all $t \in \mathcal{T}_\mathcal{I}$, i.e., that every cluster has either no children or at least two, a simple induction yields*

$$|\mathcal{T}_\mathcal{I}| \le 2|\mathcal{L}_\mathcal{I}| - 1 \le 2|\mathcal{L}_\mathcal{I}|.$$

Since the index set of every leaf cluster has to contain at least one index, we have $|\mathcal{L}_\mathcal{I}| \le |\mathcal{I}|$ and find that the estimate of Theorem 6.9 can be bounded by

$$\frac{C_{sp}+2}{2}\max\{k, r_\mathcal{I}, r_\mathcal{J}\}(|\mathcal{I}| + |\mathcal{J}|) + (C_{sp}+2)k^2(|\mathcal{I}| + |\mathcal{J}|),$$

i.e., the storage complexity of the $\mathcal{H}^2$-matrix representation is in $\mathcal{O}(\hat{k}^2(n_\mathcal{I} + n_\mathcal{J}))$, where $\hat{k} := \max\{k, r_\mathcal{I}, r_\mathcal{J}\}$ denotes the maximal rank of leaves and $n_\mathcal{I} := |\mathcal{I}|$ and $n_\mathcal{J} := |\mathcal{J}|$.

If we can guarantee $|\hat{t}| \geq k$ for all leaf cluster $t \in \mathcal{L}_{\mathcal{I}}$ by stopping the construction of the cluster tree not too late, we find

$$k|\mathcal{L}_{\mathcal{I}}| = \sum_{t\in\mathcal{L}_{\mathcal{I}}} k \leq \sum_{t\in\mathcal{L}_{\mathcal{I}}} |\hat{t}| = \Big|\bigcup_{t\in\mathcal{L}_{\mathcal{I}}} \hat{t}\Big| = |\mathcal{I}|$$

via Corollary 3.19 and conclude $|\mathcal{L}_{\mathcal{I}}| \leq |\mathcal{I}|/k$ and therefore $|\mathcal{T}_{\mathcal{I}}| \leq 2|\mathcal{I}|/k$. This gives us the improved estimate

$$\frac{3}{2}(C_{sp}+2)\max\{k, r_{\mathcal{I}}, r_{\mathcal{J}}\}(|\mathcal{I}|+|\mathcal{J}|),$$

i.e., the storage complexity is in $\mathcal{O}(\hat{k}(n_{\mathcal{I}}+n_{\mathcal{J}}))$.

Requiring that the block tree is *strictly* admissible is sometimes inconvenient, since it requires us to handle special cases if not all leaf clusters are on the same level and we have to divide blocks in only rows or columns, but not both.

If the clusters are not too irregular, we can use a weaker assumption and still obtain linear complexity.

Lemma 6.11 (Nearfield matrices) *Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be admissible and C_{sp}-sparse, and let C_{nb} be a constant such that*

$$|\hat{t}| \leq C_{nb}|\hat{s}| \qquad \text{for all } b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}} \text{ with } s\in\mathcal{L}_{\mathcal{J}}, \tag{6.5a}$$

$$|\hat{s}| \leq C_{nb}|\hat{t}| \qquad \text{for all } b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}} \text{ with } t\in\mathcal{L}_{\mathcal{I}}. \tag{6.5b}$$

The nearfield matrices $(N_b)_{b\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}}$ require not more than

$$C_{sp}C_{nb}(r_{\mathcal{I}}|\mathcal{I}| + r_{\mathcal{J}}|\mathcal{J}|) \text{ units of storage.}$$

Proof. Let $b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$. Since $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is admissible, we have $t\in\mathcal{L}_{\mathcal{I}}$ or $s\in\mathcal{L}_{\mathcal{J}}$.

If $t\in\mathcal{L}_{\mathcal{I}}$ holds, we find $|\hat{t}|\leq r_{\mathcal{I}}$ and can use (6.5b) to get $|\hat{s}|\leq C_{\text{nb}}|\hat{t}|\leq C_{\text{nb}}r_{\mathcal{I}}$.

If $s\in\mathcal{L}_{\mathcal{J}}$ holds, we find $|\hat{s}|\leq r_{\mathcal{J}}$ and can use (6.5a) to get $|\hat{t}|\leq C_{\text{nb}}|\hat{s}|\leq C_{\text{nb}}r_{\mathcal{J}}$.

The total number of coefficients is bounded by

$$\begin{aligned}
\sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} |\hat{t}|\,|\hat{s}| &\leq \sum_{\substack{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}\\ t\in\mathcal{L}_{\mathcal{I}}}} |\hat{t}|\,|\hat{s}| + \sum_{\substack{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}\\ s\in\mathcal{L}_{\mathcal{J}}}} |\hat{t}|\,|\hat{s}| \\
&\leq \sum_{t\in\mathcal{L}_{\mathcal{I}}}\sum_{s\in\text{row}(t)} |\hat{t}|\,|\hat{s}| + \sum_{s\in\mathcal{L}_{\mathcal{J}}}\sum_{t\in\text{col}(s)} |\hat{t}|\,|\hat{s}| \\
&\leq \sum_{t\in\mathcal{L}_{\mathcal{I}}}\sum_{s\in\text{row}(t)} |\hat{t}|\,C_{\text{nb}}r_{\mathcal{I}} + \sum_{s\in\mathcal{L}_{\mathcal{J}}}\sum_{t\in\text{col}(s)} |\hat{s}|\,C_{\text{nb}}r_{\mathcal{J}} \\
&\leq C_{\text{sp}}C_{\text{nb}}r_{\mathcal{I}}\sum_{t\in\mathcal{L}_{\mathcal{I}}}|\hat{t}| + C_{\text{sp}}C_{\text{nb}}r_{\mathcal{J}}\sum_{s\in\mathcal{L}_{\mathcal{J}}}|\hat{s}| \\
&\leq C_{\text{sp}}C_{\text{nb}}r_{\mathcal{I}}|\mathcal{I}| + C_{\text{sp}}C_{\text{nb}}r_{\mathcal{J}}|\mathcal{J}|,
\end{aligned}$$

where we have again used Corollary 3.19 in the last step. ■

6.3 Matrix-vector multiplication

A nested representation of a cluster basis $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ allows us to significantly reduce the corresponding storage requirements, but it also means that we can not longer work with the matrices V_t directly if t is not a leaf.

Given an admissible leaf $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, evaluating the corresponding submatrix

$$G|_{\hat{t}\times\hat{s}} = V_t S_b W_s^*$$

requires algorithms for evaluating $W_s^* x|_{\hat{s}}$ for a given vector $x \in \mathbb{K}^\mathcal{I}$ and for evaluting $V_t\hat{y}_t$ for a given vector $\hat{y}_t \in \mathbb{K}^{K_t}$.

Forward transformation. Let $W = (W_s)_{s\in\mathcal{T}_\mathcal{J}}$ be a cluster basis of rank k, and let and let $(F_s)_{s\in\mathcal{T}_\mathcal{J}}$ be a family of corresponding transfer matrices. Let $x \in \mathbb{K}^\mathcal{J}$. Our task is to compute the vectors

$$\hat{x}_s := W_s^* x|_{\hat{s}} \qquad\qquad \text{for all } s \in \mathcal{T}_\mathcal{J}. \tag{6.6}$$

This is a departure from the procedure applied for $\mathcal{H}$-matrices, where each block can essentially be treated independently from all orthers. For $\mathcal{H}^2$-matrices, the efficiency can be improved significantly by preparing suitable vectors, and in more sophisticated algorithms also matrices, in advance and share them among multiple blocks.

Let $s \in \mathcal{T}_\mathcal{J}$. If s is a leaf, we can compute (6.6) directly. If s is not a leaf, we let $n := |\operatorname{chil}(s)|$ and denote the children of s by $\operatorname{chil}(s) = \{s_1, \ldots, s_n\}$. Due to (3.12b), (3.12c) we have $\hat{s} = \hat{s}_1 \dot\cup \ldots \dot\cup \hat{s}_n$, and (6.3) yields

$$W_s = \begin{pmatrix} W_{s_1}F_{s_1} \\ \vdots \\ W_{s_n}F_{s_n} \end{pmatrix}.$$

Using this equation, we can rewrite (6.6) as

$$\begin{aligned} \hat{x}_s = W_s^* x &= \begin{pmatrix} F_{s_1}^* W_{s_1}^* & \ldots & F_{s_n}^* W_{s_n}^* \end{pmatrix} \begin{pmatrix} x|_{\hat{s}_1} \\ \vdots \\ x|_{\hat{s}_n} \end{pmatrix} \\ &= \sum_{i=1}^n F_{s_i}^* W_{s_i}^* x|_{\hat{s}_i} = \sum_{s'\in\operatorname{chil}(s)} F_{s'}^* W_{s'}^* x|_{\hat{s}'} = \sum_{s'\in\operatorname{chil}(s)} F_{s'}^* \hat{x}_{s'}. \end{aligned}$$

This means that if we ensure that the vectors $\hat{x}_{s'}$ corresponding to the children $s' \in \operatorname{chil}(s)$ are computed first, the vector $\hat{x}_s$ can be computed very efficiently using only transfer matrices.

Since we need to treat the children before their parent, a recursive algorithm is the obvious choice. It is given in Figure 6.1, and we call it the *forward transformation* for the cluster basis W.

```
procedure forward_clusterbasis($W$, $s$, $x$, var $\hat{x}$);
  if $\mathrm{chil}(s) = \emptyset$ then
    $\hat{x}_s \leftarrow W_s^* x|_{\hat{s}}$;
  else begin
    $\hat{x}_s \leftarrow 0$;
    for $s' \in \mathrm{chil}(s)$ do begin
      forward_clusterbasis($W$, $s'$, $x$, $\hat{x}$);
      $\hat{x}_s \leftarrow \hat{x}_s + F_{s'}^* \hat{x}_{s'}$
    end
  end
```

Figure 6.1: Forward transformation, $\hat{x}_r \leftarrow W_r^* x|_{\hat{r}}$ for all $r \in \mathcal{T}_s$

Coupling phase. Once the vectors $\hat{x}_s = W_s^* x|_{\hat{s}}$ are available for all clusters $s \in \mathcal{T}_{\mathcal{J}}$, we can consider the multiplication by S_b for admissible blocks $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. Instead of treating all blocks individually, we can take advantage of

$$\sum_{\substack{s\in\mathcal{T}_{\mathcal{J}}\\ b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}} V_t S_b W_s^* x|_{\hat{s}} = V_t \sum_{\substack{s\in\mathcal{T}_{\mathcal{J}}\\ b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}} S_b \hat{x}_s \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}}$$

to first compute auxiliary vectors

$$\hat{y}_t := \sum_{\substack{s\in\mathcal{T}_{\mathcal{J}}\\ b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}} S_b \hat{x}_s \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}} \tag{6.7}$$

and then perform the updates

$$y|_{\hat{t}} \leftarrow y|_{\hat{t}} + V_t \hat{y}_t \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}}. \tag{6.8}$$

The first task is straightforward: we initialize the vectors $\hat{y}_t \leftarrow 0$ for all $t \in \mathcal{T}_{\mathcal{I}}$ and then simply pass through all admissible blocks of the $\mathcal{H}^2$-matrix representation and accumulate their contributions. This is called the *coupling phase* of the $\mathcal{H}^2$-matrix-vector multiplication.

Since this phase requires us to traverse the entire blocktree, we can also handle the nearfield blocks directly via

$$y|_{\hat{t}} \leftarrow y|_{\hat{t}} + G|_{\hat{t}\times\hat{s}} x|_{\hat{s}} = y|_{\hat{t}} + N_b x|_{\hat{s}} \qquad \text{for all } b = (t,s) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}.$$

If we traverse the block tree recursively, passing through non-leaf blocks and treating leaf blocks appropriately, we arrive at the algorithm given in Figure 6.2.

The coupling phase is typically the most time-consuming part of the matrix-vector multiplication, since on the one hand the number of blocks typically far exceeds the number of clusters and on the other hand reading the coupling and nearfield matrices from storage taxes the memory interface of processors.

```
procedure fastaddeval_h2matrix($\alpha$, $G$, $b = (t,s)$, $x$, $\hat{x}$, var $y$, $\hat{y}$);
  if $b \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ then
    $\hat{y}_t \leftarrow \hat{y}_t + \alpha S_b \hat{x}_s$;
  else if $b \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$ then
    $y|_{\hat{t}} \leftarrow y|_{\hat{t}} + \alpha N_b x|_{\hat{s}}$;
  else for $b' \in \operatorname{chil}(b)$ do
    fastaddeval_h2matrix($\alpha$, $G$, $b'$, $x$, $\hat{x}$, $y$, $\hat{y}$)
```

Figure 6.2: Coupling phase, $\hat{y}_t \leftarrow \hat{y}_t + \alpha S_b \hat{x}_s$ for all admissible leaves $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ and $y|_{\hat{t}} \leftarrow y|_{\hat{t}} + \alpha N_b \hat{x}|_{\hat{s}}$ for all inadmissible leaves $b = (t,s) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$.

Backward transformation. This leaves us only with the task of performing the updates (6.8) for all $t \in \mathcal{T}_{\mathcal{I}}$. Let $t \in \mathcal{T}_{\mathcal{I}}$. If t is a leaf, we can handle (6.8) directly. If t is not a leaf, we again let $n := |\operatorname{chil}(t)|$ and denote the children of t by $\operatorname{chil}(t) = \{t_1, \ldots, t_n\}$. (6.3) yields

$$V_t = \begin{pmatrix} V_{t_1} E_{t_1} \\ \vdots \\ V_{t_n} E_{t_n} \end{pmatrix}$$

and we obtain

$$y|_{\hat{t}} \leftarrow y|_{\hat{t}} + V_t \hat{y}_t = \begin{pmatrix} y|_{\hat{t}_1} + V_{t_1} E_{t_1} \hat{y}_t \\ \vdots \\ y|_{\hat{t}_n} + V_{t_n} E_{t_n} \hat{y}_t \end{pmatrix} = \begin{pmatrix} y|_{\hat{t}_1} + V_{t_1} \tilde{y}_{t_1} \\ \vdots \\ y|_{\hat{t}_n} + V_{t_n} \tilde{y}_{t_n} \end{pmatrix}, \tag{6.9}$$

where we let

$$\tilde{y}_{t'} := E_{t'} \hat{y}_t \qquad \text{for all } t' \in \operatorname{chil}(t).$$

If we work on t before its children, we can take advantage of

$$y|_{\hat{t}'} \leftarrow y|_{\hat{t}'} + V_{t'} \hat{y}_{t'} + V_{t'} \tilde{y}_{t'} = y|_{\hat{t}'} + V_{t'}(\hat{y}_{t'} + \tilde{y}_{t'}) \qquad \text{for all } t' \in \operatorname{chil}(t)$$

to simply add $\tilde{y}_{t'}$ to the vectors $\hat{y}_{t'}$ corresponding to the sons. As long as we have no further need for the vectors $\hat{y}_{t'}$, we can simply update them.

The resulting algorithm is given in Figure 6.3, and we call it the *backward transformation* for the cluster basis V.

Both the forward and the backward transformation algorithms take advantage of the fact that we have to perform operations for *all* clusters: in the forward transformation, we can re-use the intermediate results $\hat{x}_{s'}$ computed for the children $s' \in \operatorname{chil}(s)$, while in the backward transformation we can shift work from the parent t to its children $t' \in \operatorname{chil}(t)$. Even if we wanted to compute only $\hat{x}_s$ for *one* cluster, we still would have to compute $\hat{x}_{s'}$ for all its descendants.

```
procedure backward_clusterbasis($V$, $t$, var $\hat{y}$, $y$);
  if $\operatorname{chil}(t) = \emptyset$ then
    $y|_{\hat{t}} \leftarrow y|_{\hat{t}} + V_t \hat{y}_t$;
  else
    for $t' \in \operatorname{chil}(t)$ do begin
      $\hat{y}_{t'} \leftarrow \hat{y}_{t'} + E_{t'} \hat{y}_t$;
      backward_clusterbasis($V$, $t'$, $\hat{y}$, $y$);
    end
  end
```

Figure 6.3: Backward transformation, $y|_{\hat{r}} \leftarrow y|_{\hat{r}} + V_r \hat{y}_r$ for all $r \in \mathcal{T}_t$. The coefficients $(\hat{y}_r)_{t \in \mathcal{T}_t}$ are overwritten by intermediate results.

```
procedure addeval_h2matrix($\alpha$, $G$, $b = (t, s)$, $x$, var $y$);
  forward_clusterbasis($W$, $s$, $x$, $\hat{x}$);
  $\hat{y} \leftarrow 0$;
  fastaddeval_h2matrix($\alpha$, $G$, $b$, $x$, $\hat{x}$, $y$, $\hat{y}$);
  backward_clusterbasis($V$, $t$, $\hat{y}$, $y$)
```

Figure 6.4: $\mathcal{H}^2$-matrix-vector multiplication, $y|_{\hat{t}} \leftarrow y|_{\hat{t}} + \alpha G|_{\hat{t}\times\hat{s}} x|_{\hat{s}}$.

Using the forward and the backward transformation, we can construct an efficient algorithm for performing the matrix-vector multiplication $y \leftarrow y + \alpha G x$ with an $\mathcal{H}^2$-matrix G in $\mathcal{H}^2$-matrix representation. The forward transformation yields the auxiliary vectors $\hat{x}_s$, the coupling phase prepares the vectors $\hat{y}_t$, and the backward transformation adds them to the final result. The algorithm is summarized in Figure 6.4.

Remark 6.12 (Adjoint) *To perform the adjoint matrix-vector multiplication $x \leftarrow x + \alpha G^* y$, we can apply the forward transformation to the row basis V, switch to the adjoint matrices S_b^* and N_b^* in the coupling phase, and apply the backward transformation to the colum basis W.*

Remark 6.13 (Recursion) *Using recursive algorithms for the forward and backward transformation and the coupling phase is convenient, particularly if we expect to apply the algorithms also to submatrices of an $\mathcal{H}^2$-matrix, e.g., in the course of arithmetic operations like the matrix multiplication or inversion.*

If we want to parallelize the algorithms, different approaches may be more appropriate, e.g., the forward and backward transformation could be performed level by level, such that all clusters on the same level can be treated in parallel.

For the coupling phase, it may be a good idea to use a description of the sets $\operatorname{row}(t)$ *and* $\operatorname{col}(s)$*, e.g., as a list of blocks, in order to avoid write conflicts caused by different blocks trying to update the same vector.*

6.4 Low-rank structure of $\mathcal{H}^2$-matrices

We have seen that a matrix can be approximated by an $\mathcal{H}$-matrix if all submatrices corresponding to admissible blocks can be approximated by low-rank matrices. We will now characterize $\mathcal{H}^2$-matrices in terms of ranks of suitable submatrices.

Since we are only interested in *admissible* blocks, we restrict the sets $\operatorname{row}(t)$ and $\operatorname{col}(s)$ introduced in Definition 3.31 to

$$\operatorname{row}^+(t) := \{s \in \mathcal{T}_{\mathcal{J}} \ : \ (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}\} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}},$$
$$\operatorname{col}^+(s) := \{t \in \mathcal{T}_{\mathcal{I}} \ : \ (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}\} \qquad \text{for all } s \in \mathcal{T}_{\mathcal{J}}.$$

We consider a row cluster $t \in \mathcal{T}_{\mathcal{I}}$ and define the number of admissible blocks connected to it by $n := |\operatorname{row}^+(t)|$ and the corresponding column clusters by $\operatorname{row}^+(t) = \{s_1, \ldots, s_n\}$. Due to (6.4), we have $G|_{\hat{t}\times\hat{s}_i} = V_t S_{t,s_i} W_{s_i}^*$ for all $i \in [1:n]$ and therefore

$$G_t := \begin{pmatrix} G|_{\hat{t}\times\hat{s}_1} & \ldots & G|_{\hat{t}\times\hat{s}_n} \end{pmatrix} = V_t \begin{pmatrix} S_{t,s_1} W_{s_1}^* & \ldots & S_{t,s_n} W_{s_n}^* \end{pmatrix}.$$

We have found a factorized representation of G_t, so its rank cannot exceed k. Since the cluster bases are nested, we can extend this argument to larger matrices G_t including blocks connected to the predecessors of t, as well.

Lemma 6.14 (Extended transfer matrices) *Let $V = (V_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ be a cluster basis of rank k, and let $(E_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ be a corresponding family of transfer matrices.*

We define the extended transfer matrices *inductively by*

$$E_{r,t} := \begin{cases} I & \text{if } r = t, \\ E_{r,t'} E_{t'} & \text{if } r \in \mathcal{T}_{t'},\ t' \in \operatorname{chil}(t) \end{cases} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}},\ r \in \mathcal{T}_t.$$

We have

$$V_t|_{\hat{r}} = V_r E_{r,t} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}},\ r \in \mathcal{T}_t.$$

Proof. By induction on $\operatorname{level}(r) - \operatorname{level}(t)$.

Let $t \in \mathcal{T}_{\mathcal{I}}$ and $r \in \mathcal{T}_t$ with $\operatorname{level}(r) - \operatorname{level}(t) = 0$. Then we have $r = t$, and our definition yields

$$V_t|_{\hat{r}} = V_t = V_r = V_r E_{r,r} = V_t E_{r,t}.$$

Let now $n \in \mathbb{N}_0$ be given such that our claim holds for all $t \in \mathcal{T}_{\mathcal{I}}$ and $r \in \mathcal{T}_t$ with $\operatorname{level}(r) - \operatorname{level}(t) = n$.

Let $t \in \mathcal{T}_{\mathcal{I}}$ and $r \in \mathcal{T}_t$ with $\operatorname{level}(r) - \operatorname{level}(t) = n+1$. Due to $\operatorname{level}(t) < \operatorname{level}(r)$, we can find $t' \in \operatorname{chil}(t)$ such that $r \in \mathcal{T}_{t'}$. Due to $\operatorname{level}(r) - \operatorname{level}(t') = \operatorname{level}(r) - (\operatorname{level}(t) + 1) = n + 1 - 1 = n$, we can apply the induction assumption to find

$$V_{t'}|_{\hat{r}} = V_r E_{r,t'},$$

and (6.3) with $\hat{r} \subseteq \hat{t}'$ yields

$$V_t|_{\hat{r}} = (V_t|_{\hat{t}'})|_{\hat{r}} = (V_{t'} E_{t'})|_{\hat{r}} = V_{t'}|_{\hat{r}} E_{t'} = V_r E_{r,t'} E_{t'} = V_r E_{r,t}.$$

∎

Lemma 6.15 (Low-rank structure) *Let $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ be an $\mathcal{H}^2$-matrix with row cluster basis V and column cluster basis W of rank k. The sets*

$$\mathcal{F}_t := \bigcup_{t_\top \in \operatorname{pred}(t)} \bigcup_{\substack{s\in\mathcal{T}_\mathcal{J} \\ (t_\top,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}} \hat{s} \subseteq \mathcal{J} \qquad \text{for all } t \in \mathcal{T}_\mathcal{I}, \tag{6.10a}$$

$$\mathcal{F}_s := \bigcup_{s_\top \in \operatorname{pred}(s)} \bigcup_{\substack{t\in\mathcal{T}_\mathcal{I} \\ (t,s_\top)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}} \hat{t} \subseteq \mathcal{I} \qquad \text{for all } s \in \mathcal{T}_\mathcal{J} \tag{6.10b}$$

are called the farfield indices *for t and s.*

For every $j \in \mathcal{F}_t$, there are exactly one $t_\top \in \operatorname{pred}(t)$ and exactly one $s \in \mathcal{T}_\mathcal{J}$ with $(t_\top, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ such that $j \in \hat{s}$.

For every $i \in \mathcal{F}_s$, there are exactly one $s_\top \in \operatorname{pred}(s)$ and exactly one $t \in \mathcal{T}_\mathcal{I}$ with $(t, s_\top) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ such that $i \in \hat{t}$.

For all $t \in \mathcal{T}_\mathcal{I}$, we can find a matrix $B_t \in \mathbb{K}^{\mathcal{F}_t\times k}$ such that

$$G|_{\hat{t}\times\mathcal{F}_t} = V_t B_t^*,$$

and for all $s \in \mathcal{T}_\mathcal{J}$, we can find a matrix $A_s \in \mathbb{K}^{\mathcal{F}_s\times k}$ such that

$$G|_{\mathcal{F}_s\times\hat{s}} = A_s W_s^*.$$

Proof. Let $t \in \mathcal{T}_\mathcal{I}$. We first prove that for each $j \in \mathcal{F}_t$, there are exactly one $t_\top \in \operatorname{pred}(t)$ and exactly one $s \in \mathcal{T}_\mathcal{J}$ with $(t_\top, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ such that $j \in \hat{s}$.

Let $j \in \mathcal{F}_t$, and let $t_{\top,1}, t_{\top,2} \in \operatorname{pred}(t)$, $s_1, s_2 \in \mathcal{T}_\mathcal{J}$ with $(t_{\top,1}, s_1) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ and $(t_{\top,2}, s_2) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ such that $j \in \hat{s}_1$ and $j \in \hat{s}_2$.

Let $i \in \hat{t}$. Then we have $i \in \hat{t}_{\top,1}$ and $i \in \hat{t}_{\top,2}$ and therefore $(i,j) \in \hat{t}_{\top,1} \times \hat{s}_1$ and $(i,j) \in \hat{t}_{\top,2} \times \hat{s}_2$. Due to Corollary 3.23, this already implies both $t_{\top,1} = t_{\top,2}$ and $s_1 = s_2$.

Let $j \in \mathcal{F}_t$, and let $t_\top \in \operatorname{pred}(t)$ and $s \in \operatorname{row}(t_\top)$ be the unique clusters defined above. By Definition 6.5, we have

$$G|_{\hat{t}_\top\times\hat{s}} = V_{t_\top} S_b W_s^*,$$

and Lemma 6.14 yields

$$G|_{\hat{t}\times\hat{s}} = (G|_{\hat{t}_\top\times\hat{s}})|_{\hat{t}\times\hat{s}} = V_{t_\top}|_{\hat{t}} S_b W_s^* = V_t E_{t,t_\top} S_b W_s^*.$$

We conclude that

$$B_t|_{\hat{s}\times K_t} := W_s S_b^* E_{t,t_\top}^* \qquad \text{for all } t_\top \in \operatorname{pred}(t),\ s \in \operatorname{row}(t_\top) \text{ with } b = (t_\top, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$$

defines the matrix $B_t \in \mathbb{K}^{\mathcal{F}_t\times K_t}$ uniquely, and we have

$$G|_{\hat{t}\times\hat{s}} = V_t E_{t,t_\top} S_b W_s^* = V_t B_t|_{\hat{s}}^* \qquad \text{for all } t_\top \in \operatorname{pred}(t),\ s \in \operatorname{row}(t_\top) \text{ with } b = (t_\top, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}.$$

The corresponding results for column clusters can be obtained by similar arguments. ∎

Lemma 6.16 (Partial inverse) *Let $V \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$. Then there is a matrix $R \in \mathbb{K}^{\mathcal{J}\times\mathcal{I}}$ such that $VRV = V$.*

Proof. Let $k := \dim(\operatorname{range}(V))$, and let $\{v_1, \ldots, v_k\}$ be a basis of $\operatorname{range}(V)$. We can extend it to a basis $\{v_1, \ldots, v_n\}$ of $\mathbb{K}^{\mathcal{I}}$, where $n := \dim(\mathbb{K}^{\mathcal{I}}) = |\mathcal{I}|$.

For each $i \in [1 : k]$, we can find a vector $x_i \in \mathbb{K}^{\mathcal{J}}$ such that $Vx_i = v_i$.

Since $\{v_1, \ldots, v_n\}$ is a basis of $\mathbb{K}^{\mathcal{I}}$, we can define $R \in \mathbb{K}^{\mathcal{J}\times\mathcal{I}}$ by

$$Rv_i := \begin{cases} x_i & \text{if } i \leq k, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i \in [1 : n]$$

and obtain

$$VRv_i = Vx_i = v_i \qquad \text{for all } i \in [1 : k].$$

Let now $y \in \mathbb{K}^{\mathcal{J}}$. Since $\{v_1, \ldots, v_k\}$ is a basis of $\operatorname{range}(V)$, we can find $\alpha_1, \ldots, \alpha_k \in \mathbb{K}$ such that

$$Vy = \alpha_1 v_1 + \ldots + \alpha_k v_k.$$

This implies

$$VRVy = \alpha_1 VRv_1 + \ldots + \alpha_k VRv_k = \alpha_1 v_1 + \ldots + \alpha_k v_k = Vy,$$

and we have proven $VRV = V$. ∎

Corollary 6.17 (Characterization via bases) *Let $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$, let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a block tree, and let V and W be cluster bases of rank k.*

G is an $\mathcal{H}^2$-matrix with row basis V and column basis W if and only if for all $t \in \mathcal{T}_{\mathcal{I}}$ and $s \in \mathcal{T}_{\mathcal{J}}$ there are matrices $A_s \in \mathbb{K}^{\mathcal{F}_s\times k}$ and $B_t \in \mathbb{K}^{\mathcal{F}_t\times k}$ such that

$$G|_{\mathcal{F}_s\times\hat{s}} = A_s W_s^*, \qquad G|_{\hat{t}\times\mathcal{F}_t} = V_t B_t^*. \tag{6.11}$$

Proof. If G is an $\mathcal{H}^2$-matrix, we can apply Lemma 6.15 to obtain A_s and B_t satisfying the equations (6.11.

Assume now that for all $t \in \mathcal{T}_{\mathcal{I}}$ and $s \in \mathcal{T}_{\mathcal{J}}$ there are $A_s \in \mathbb{K}^{\mathcal{F}_s\times k}$ and $B_t \in \mathbb{K}^{\mathcal{F}_t\times k}$ such that (6.11) holds. In order to prove that G is an $\mathcal{H}^2$-matrix, we have to construct coupling matrices $S_b \in \mathbb{K}^{k\times k}$ for all admissible leaves $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$.

Let $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. By definition, we have $\hat{s} \subseteq \mathcal{F}_t$ and $\hat{t} \subseteq \mathcal{F}_s$, and (6.11) yields

$$A_s|_{\hat{t}} W_s^* = (A_s W_s^*)|_{\hat{t}\times\hat{s}} = G|_{\hat{t}\times\hat{s}} = (V_t B_t^*)|_{\hat{t}\times\hat{s}} = V_t B_t|_{\hat{s}}^*.$$

Lemma 6.16 yields a matrix $R \in \mathbb{K}^{k\times\hat{t}}$ such that $V_t R V_t = V_t$, and we obtain

$$G|_{\hat{t}\times\hat{s}} = V_t B_t|_{\hat{s}}^* = V_t R V_t B_t|_{\hat{s}}^* = V_t R G|_{\hat{t}\times\hat{s}} = V_t R A_s|_{\hat{t}} W_s^* = V_t S_b W_s^*,$$

where we let $S_b := RA_s|_{\hat{t}\times k}$. We have proven (6.4), so G has to be an $\mathcal{H}^2$-matrix. ∎

Theorem 6.18 (Characterization via ranks) *Let $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$, let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be a block tree, and let $\mathcal{F}_t$ and $\mathcal{F}_s$ for all $t \in \mathcal{T}_\mathcal{I}$ and $s \in \mathcal{T}_\mathcal{J}$ be defined as in Lemma 6.15.*

G is an $\mathcal{H}^2$-matrix with row and column cluster bases of rank k if and only if

$$\operatorname{rank}(G|_{\hat{t}\times\mathcal{F}_t}) \leq k, \qquad \operatorname{rank}(G|_{\mathcal{F}_s\times\hat{s}}) \leq k \qquad \text{for all } t \in \mathcal{T}_\mathcal{I},\ s \in \mathcal{T}_\mathcal{J}.$$

Proof. If G is an $\mathcal{H}^2$-matrix, our claim is a direct consequence of Corollary 6.17.

We assume

$$\operatorname{rank}(G|_{\hat{t}\times\mathcal{F}_t}) \leq k \qquad \text{for all } t \in \mathcal{T}_\mathcal{I}$$

and are looking for a row cluster basis satisfying the second equation in (6.11).

We will prove by induction on $|\mathcal{T}_t|$ that for every $t \in \mathcal{T}_\mathcal{I}$ there are $V_t \in \mathbb{K}^{\hat{t}\times k}$ and $B_t \in \mathbb{K}^{\mathcal{F}_t\times k}$ such that

$$G|_{\hat{t}\times\mathcal{F}_t} = V_t B_t^*$$

and that if $\operatorname{chil}(t) \neq \emptyset$ holds there are transfer matrices $E_{t'} \in \mathbb{K}^{k\times k}$ for all $t' \in \operatorname{chil}(t)$ such that

$$V_t|_{\hat{t}'} = V_{t'} E_{t'} \qquad \text{for all } t' \in \operatorname{chil}(t).$$

Let $t \in \mathcal{T}_\mathcal{I}$ with $|\mathcal{T}_t| = 1$, i.e., $\operatorname{chil}(t) = \emptyset$. Due to $\operatorname{rank}(G|_{\hat{t}\times\mathcal{F}_t}) \leq k$, we can apply Corollary 4.9 yields matrices $V_t \in \mathbb{K}^{\hat{t}\times k}$ and $B_t \in \mathbb{K}^{\mathcal{F}_t\times k}$ with

$$G|_{\hat{t}\times\mathcal{F}_t} = V_t B_t^*.$$

Let now $n \in \mathbb{N}$ be given such that for every $t \in \mathcal{T}_\mathcal{I}$ with $|\mathcal{T}_t| \leq n$ matrices $V_t \in \mathbb{K}^{\hat{t}\times k}$ and $B_t \in \mathbb{K}^{\mathcal{F}_t\times k}$ with $G|_{\hat{t}\times\mathcal{F}_t} = V_t B_t^*$ exist.

Let $t \in \mathcal{T}_\mathcal{I}$ with $|\mathcal{T}_t| = n+1$. This implies $\operatorname{chil}(t) \neq \emptyset$. We let $m := |\operatorname{chil}(t)|$ and $\operatorname{chil}(t) = \{t_1, \ldots, t_m\}$.

Since every $t' \in \operatorname{chil}(t)$ satisfies $|\mathcal{T}_{t'}| \leq n$, we can use the induction assumption to find $V_{t'} \in \mathbb{K}^{\hat{t}'\times k}$ and $B_{t'} \in \mathbb{K}^{\mathcal{F}_{t'}\times k}$ such that $G|_{\hat{t}'\times\mathcal{F}_{t'}} = V_{t'} B_{t'}^*$.

Using Lemma 6.16, we find matrices $R_{t'} \in \mathbb{K}^{k\times\hat{t}'}$ with $V_{t'} R_{t'} V_{t'} = V_{t'}$ for all $t' \in \operatorname{chil}(t)$.

Our definition implies $\mathcal{F}_t \subseteq \mathcal{F}_{t'}$ for all $t' \in \operatorname{chil}(t)$, and we find

$$\begin{aligned}
G|_{\hat{t}\times\mathcal{F}_t} &= \begin{pmatrix} G|_{\hat{t}_1\times\mathcal{F}_t} \\ \vdots \\ G|_{\hat{t}_m\times\mathcal{F}_t} \end{pmatrix} = \begin{pmatrix} (V_{t_1}B_{t_1}^*)|_{\hat{t}_1\times\mathcal{F}_t} \\ \vdots \\ (V_{t_m}B_{t_m}^*)|_{\hat{t}_m\times\mathcal{F}_t} \end{pmatrix} = \begin{pmatrix} V_{t_1}R_{t_1}V_{t_1}B_{t_1}|_{\mathcal{F}_t}^* \\ \vdots \\ V_{t_m}R_{t_m}V_{t_m}B_{t_m}|_{\mathcal{F}_t}^* \end{pmatrix} \\
&= \begin{pmatrix} V_{t_1} & & \\ & \ddots & \\ & & V_{t_m} \end{pmatrix} \begin{pmatrix} R_{t_1} & & \\ & \ddots & \\ & & R_{t_m} \end{pmatrix} \begin{pmatrix} V_{t_1}B_{t_1}|_{\mathcal{F}_t}^* \\ \vdots \\ V_{t_m}B_{t_m}|_{\mathcal{F}_t}^* \end{pmatrix} \\
&= \begin{pmatrix} V_{t_1} & & \\ & \ddots & \\ & & V_{t_m} \end{pmatrix} \underbrace{\begin{pmatrix} R_{t_1} & & \\ & \ddots & \\ & & R_{t_m} \end{pmatrix} G|_{\hat{t}\times\mathcal{F}_t}}_{=:\widehat{G}_t}.
\end{aligned}$$

Due to $\operatorname{rank}(G|_{\hat{t}\times\mathcal{F}_t}) \leq k$, we also have $\operatorname{rank}(\widehat{G}_t) \leq k$ and can again apply Corollary 4.9 to find matrices $\widehat{V}_t \in \mathbb{K}^{(mk)\times k}$ and $B_t \in \mathbb{K}^{\mathcal{F}_t\times k}$ with $\widehat{G}_t = \widehat{V}_t B_t^*$.

Combining with the previous equation we obtain

$$G|_{\hat{t}\times\mathcal{F}_t} = \begin{pmatrix} V_{t_1} & & \\ & \ddots & \\ & & V_{t_m} \end{pmatrix} \widehat{G}_t = \begin{pmatrix} V_{t_1} & & \\ & \ddots & \\ & & V_{t_m} \end{pmatrix} \widehat{V}_t B_t^*,$$

and splitting $\widehat{V}_t$ into $k \times k$ submatrices yields

$$\widehat{V}_t = \begin{pmatrix} E_{t_1} \\ \vdots \\ E_{t_m} \end{pmatrix}.$$

We define

$$V_t := \begin{pmatrix} V_{t_1}E_{t_1} \\ \vdots \\ V_{t_m}E_{t_m} \end{pmatrix} \in \mathbb{K}^{\hat{t}\times k} \tag{6.12}$$

and conclude

$$G|_{\hat{t}\times\mathcal{F}_t} = \begin{pmatrix} V_{t_1} & & \\ & \ddots & \\ & & V_{t_m} \end{pmatrix} \widehat{V}_t B_t^* = V_t B_t^*.$$

This is the required factorization, and (6.12) ensures that V_t is defined via transfer matrices, i.e., that we have a *nested* basis.

The same arguments can be applied to construct a column basis. ■

7 $\mathcal{H}^2$-matrix compression

While the approximation of a given matrix by an $\mathcal{H}$-matrix can be computed by considering the singular value decompositions of all admissible blocks, finding an approximation by an $\mathcal{H}^2$-matrix poses a greater challenge: cluster bases introduce dependencies between blocks, and the nested structure introduces dependencies between levels.

It turns out that these dependencies are both a blessing and a curse: they can be used to construct an algorithm of *quadratic* complexity where $\mathcal{H}$-matrices require cubic complexity, but the algorithm is a little more complicated than simple singular value decompositions.

7.1 Orthogonal projections

We have seen in Theorem 6.18 that a matrix $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ is an $\mathcal{H}^2$-matrix if and only if the submatrices $G|_{\hat{t}\times\mathcal{F}_t}$ and $G|_{\mathcal{F}_s\times\hat{s}}$ are of low rank for all $t \in \mathcal{T}_{\mathcal{I}}$ and $s \in \mathcal{T}_{\mathcal{J}}$, where $\mathcal{F}_t$ and $\mathcal{F}_s$ denote the farfields of t and s, respectively.

In order to obtain an *approximation* of an arbitrary given matrix by an $\mathcal{H}^2$-matrix, we have to find low-rank approximations of these submatrices. These approximations can be conveniently constructed by projections into the ranges of the basis matrices V_t and W_s, and the best possible projections are the *orthogonal projections*.

Definition 7.1 (Orthogonal projection) *Let $P \in \mathbb{K}^{\mathcal{I}\times\mathcal{I}}$. If $P^2 = P = P^*$ holds, we call P an* orthogonal projection.

Lemma 7.2 (Orthogonal projection) *Let $P \in \mathbb{K}^{\mathcal{I}\times\mathcal{I}}$ be an orthogonal projection. We have*

$$\|x-y\|^2 = \|x-Px\|^2 + \|y-Px\|^2 \qquad \text{for all } x \in \mathbb{K}^{\mathcal{I}},\ y \in \operatorname{range}(P). \tag{7.1}$$

In particular, we have

$$\begin{aligned}
\|x-Px\|^2 &\leq \|x-y\|^2 && \text{for all } x \in \mathbb{K}^{\mathcal{I}},\ y \in \operatorname{range}(P), && (7.2a)\\
\|x-Px\|^2 &= \|x\|^2 - \|Px\|^2 && \text{for all } x \in \mathbb{K}^{\mathcal{I}}, && (7.2b)\\
\|Px\|^2 &\leq \|x\|^2 && \text{for all } x \in \mathbb{K}^{\mathcal{I}}, && (7.2c)
\end{aligned}$$

i.e., Px is the best approximation of a vector $x \in \mathbb{K}^{\mathcal{I}}$ in the range of P, and applying the projection does not increase the norm.

Proof. Let $x \in \mathbb{K}^{\mathcal{I}}$ and $y \in \operatorname{range}(P)$. We can find $z \in \mathbb{K}^{\mathcal{I}}$ with $y = Pz$ by definition and obtain $Py = P^2 z = Pz = y$. We have

$$x - y = x - Px + Px - y = x - Px + P(x - y)$$

and find

$$\begin{aligned}
\|x - y\|^2 &= \langle x - y, x - y \rangle = \langle x - Px + P(x - y), x - Px + P(x - y) \rangle \\
&= \langle x - Px, x - Px \rangle + \langle x - Px, P(x - y) \rangle \\
&\qquad + \langle P(x - y), x - Px \rangle + \langle P(x - y), P(x - y) \rangle \\
&= \|x - Px\|^2 + \langle P^*(x - Px), x - y \rangle \\
&\qquad + \langle x - y, P^*(x - Px) \rangle + \|P(x - y)\|^2 \\
&= \|x - Px\|^2 + \langle Px - P^2 x, P(x - y) \rangle \\
&\qquad + \langle x - y, Px - P^2 x \rangle + \|Px - y\|^2 \\
&= \|x - Px\|^2 + \|Px - y\|^2.
\end{aligned}$$

We obtain (7.2a) from (7.1) by

$$\|x - Px\|^2 \leq \|x - Px\|^2 + \|y - Px\|^2 = \|x - y\|^2,$$

applying (7.1) to $y = 0$ yields $\|x - Px\|^2 + \|Px\|^2 = \|x\|^2$, which is equivalent to (7.2b) and also gives us $\|Px\|^2 \leq \|x\|^2$, i.e., (7.2c). ∎

Considering the approximation of a submatrix $G|_{\hat{t} \times \hat{s}}$ by an $\mathcal{H}^2$-matrix block $V_t S_b W_s^*$, it would be advantageous if we could construct an orthogonal projection P such that $\operatorname{range}(P) = \operatorname{range}(V_t)$.

Fortunately, *isometric* matrices (cf. Definition 4.15) immediately give rise to orthogonal projections.

Lemma 7.3 (Factorized projection) *Let $V \in \mathbb{K}^{\mathcal{I} \times \mathcal{J}}$ be an isometric matrix.*
Then $P := VV^* \in \mathbb{K}^{\mathcal{I} \times \mathcal{I}}$ is an orthogonal projection with $\operatorname{range}(P) = \operatorname{range}(V)$.

Proof. We have

$$\begin{aligned}
P^2 &= VV^*VV^* = V(V^*V)V^* = VV^* = P, \\
P^* &= (VV^*)^* = V^{**}V^* = VV^* = P,
\end{aligned}$$

so P is indeed an orthogonal projection.

We have $\operatorname{range}(P) \subseteq \operatorname{range}(V)$ by definition. Let $x \in \operatorname{range}(V)$. Then there is a $\hat{x} \in \mathbb{K}^{\mathcal{J}}$ with $x = V\hat{x}$ and we obtain

$$Px = VV^*x = VV^*V\hat{x} = V(V^*V)\hat{x} = V\hat{x} = x,$$

i.e., $x \in \operatorname{range}(P)$. ∎

Lemma 7.4 (Blockwise projection) *Let $V \in \mathbb{K}^{\mathcal{I}\times\mathcal{K}}$ and $W \in \mathbb{K}^{\mathcal{J}\times\mathcal{L}}$ be isometric matrices.*

Let $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ and $S := V^*GW \in \mathbb{K}^{\mathcal{K}\times\mathcal{L}}$. We have

$$\|G - VSW^*\|_2^2 \le \|G - VV^*G\|_2^2 + \|G - GWW^*\|_2^2. \tag{7.3a}$$

Since we also have

$$\|G - VV^*G\|_2 \le \|G - VSW^*\|_2, \qquad \|G - GWW^*\|_2 \le \|G - VSW^*\|_2, \tag{7.3b}$$

the bound (7.3a) cannot overestimate the error by a factor of more than $\sqrt{2}$.

Proof. Let $z \in \mathbb{K}^{\mathcal{J}}$. We let $P := VV^*$ and find

$$\|(G - VSW^*)z\|_2^2 = \|(G - VV^*GWW^*)z\|_2^2 = \|Gz - PGWW^*z\|_2^2.$$

We can apply (7.1) to $x := Gz$ and $y := PGWW^*z$ and obtain

$$\begin{aligned}\|(G - VSW^*)z\|_2^2 &= \|Gz - PGz\|_2^2 + \|PGWW^*z - PGz\|_2^2\\ &= \|Gz - VV^*Gz\|_2^2 + \|P(GWW^*z - Gz)\|_2^2.\end{aligned} \tag{7.4}$$

Using (7.2c) to bound the second term yields

$$\begin{aligned}\|(G - VSW^*)z\|_2^2 &\le \|Gz - VV^*Gz\|_2^2 + \|GWW^*z - Gz\|_2^2\\ &= \|(G - VV^*G)z\|_2^2 + \|(G - GWW^*)z\|_2^2.\end{aligned}$$

Since this holds for all $z \in \mathbb{K}^{\mathcal{J}}$, we conclude

$$\|G - VSW^*\|_2^2 \le \|G - VV^*G\|_2^2 + \|G - GWW^*\|_2^2. \tag{7.5}$$

The equation (7.4) implies

$$\|(G - VSW^*)z\|_2 \ge \|Gz - VV^*Gz\|_2 = \|(G - VV^*G)z\|_2,$$

and taking the supremum yields the first estimate of (7.3b).

We use (4.10c) and apply this result to G^*, W and V instead of G, V and W in order to obtain

$$\|G - GWW^*\|_2 = \|G^* - WW^*G^*\|_2 \le \|G^* - WW^*G^*VV^*\|_2 = \|G - VSW^*\|_2.$$

This is the second estimate of (7.3b). ■

Definition 7.5 (Isometric cluster basis) *Let $V = (V_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ be a cluster basis. If all basis matrices are isometric, i.e., if we have*

$$V_t^*V_t = I \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}},$$

we call V an isometric *cluster basis.*

If we have isometric cluster bases $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $W = (W_s)_{s\in\mathcal{T}_\mathcal{J}}$, we can use Lemma 7.4 to compute quasi-optimal coupling matrices

$$S_b := V_t^* G|_{\hat{t}\times\hat{s}} W_s \qquad \text{for all } b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}.$$

Due to (7.3a), we can split the approximation error into contributions by the row and column basis projections.

We can even construct the row and column bases independently: if we first find an isometric row basis $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ such that

$$\|G|_{\hat{t}\times\hat{s}} - V_t V_t^* G|_{\hat{t}\times\hat{s}}\|_2 \leq \epsilon \qquad \text{for all } b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$$

and then an isometric column basis $W = (W_s)_{s\in\mathcal{T}_\mathcal{J}}$ such that

$$\|G|^*_{\hat{t}\times\hat{s}} - W_s W_s^* G|^*_{\hat{t}\times\hat{s}}\|_2 \leq \epsilon \qquad \text{for all } b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},$$

Lemma 7.4 yields

$$\|G|_{\hat{t}\times\hat{s}} - V_t S_b W_s^*\|_2 \leq \sqrt{2}\epsilon \qquad \text{for all } b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}.$$

7.2 Adaptive cluster bases

Given a matrix $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ and a block tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, the construction of an optimal $\mathcal{H}$-matrix approximation is fairly straightforward: we compute the singular value decomposition of $G|_{\hat{t}\times\hat{s}}$ for all admissible leaves $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ and obtain the best possible low-rank approximation by removing the smallest singular values (cf. Lemma 4.25 and Theorem 4.29).

Constructing a quasi-optimal $\mathcal{H}^2$-matrix approximation of a matrix G is more of a challenge: we cannot handle individual blocks, but have to find cluster bases that are appropriate for multiple blocks spread across multiple levels of the block tree. Orthogonal projections allow us to find a relatively simple and very flexible algorithm that directly constructs a suitable isometric cluster basis in nested representation [8], [6, Chapter 6].

We consider only the row basis $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$, since Lemma 7.4 allows us to obtain the column basis $W = (W_s)_{s\in\mathcal{T}_\mathcal{J}}$ by applying the procedure to the adjoint matrix G^* with a corresponding "adjoint" block tree.

Theorem 6.18 suggests that we should look for low-rank approximations of the matrices

$$G_t := G|_{\hat{t}\times\mathcal{F}_t} \qquad \text{for all } t \in \mathcal{T}_\mathcal{I}.$$

Due to Lemma 7.3, we can expect good approximation and stability properties from an isometric cluster basis, so these low-rank approximations should be of the form

$$G_t \approx V_t V_t^* G_t \qquad \text{for all } t \in \mathcal{T}_\mathcal{I},$$

where $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ is an isometric cluster basis.

We can construct the matrices V_t by a recursive algorithm.

Case 1: t is a leaf. In this case, we compute the singular value decomposition

$$G_t = Q\Sigma P^*, \qquad \text{with } Q \in \mathbb{K}^{\hat{t}\times p},\ P \in \mathbb{K}^{\mathcal{F}_t\times p} \text{ isometric, } \Sigma = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_p \end{pmatrix}.$$

We denote the columns of Q by $q_1, \ldots, q_p \in \mathbb{K}^{\hat{t}}$, i.e., we have

$$Q = \begin{pmatrix} q_1 & \ldots & q_p \end{pmatrix}.$$

Given a rank $k_t \in [0:p]$, we let

$$V_t := \begin{pmatrix} q_1 & \ldots & q_{k_t} \end{pmatrix} \in \mathbb{K}^{\hat{t}\times k_t}$$

and observe

$$V_t V_t^* Q = \begin{pmatrix} q_1 & \ldots & q_{k_t} & 0 & \ldots & 0 \end{pmatrix}, \qquad V_t V_t^* G_t = Q \begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_{k_t} & & \\ & & & 0 & \\ & & & & \ddots \end{pmatrix} P^*.$$

Lemma 4.25 provides us with the error equation

$$\|G_t - V_t V_t^* G_t\|_2 = \begin{cases} \sigma_{k_t+1} & \text{if } k_t < p, \\ 0 & \text{otherwise} \end{cases}, \tag{7.6}$$

and Theorem 4.29 states that there is no better approximation of G_t.

Case 2: t is not a leaf. In this case, we let $n := |\operatorname{chil}(t)|$ and $\operatorname{chil}(t) = \{t_1, \ldots, t_n\}$. We first compute isometric basis matrices $V_{t_1}, \ldots, V_{t_n}$ for all children by recursion. We let

$$m_t := \sum_{t' \in \operatorname{chil}(t)} k_{t'}.$$

Since we are looking for a cluster basis, there have to be transfer matrices $E_{t_1}, \ldots, E_{t_n}$ such that

$$V_t = \begin{pmatrix} V_{t_1} & & \\ & \ddots & \\ & & V_{t_n} \end{pmatrix} \begin{pmatrix} E_{t_1} \\ \vdots \\ E_{t_n} \end{pmatrix}.$$

In order to keep the notation simple, we introduce

$$U_t := \begin{pmatrix} V_{t_1} & & \\ & \ddots & \\ & & V_{t_n} \end{pmatrix} \in \mathbb{K}^{\hat{t}\times m_t}, \qquad \widehat{V}_t := \begin{pmatrix} E_{t_1} \\ \vdots \\ E_{t_n} \end{pmatrix} \in \mathbb{K}^{m_t\times k_t} \tag{7.7}$$

and write the equation as

$$V_t = U_t \widehat{V}_t.$$

Since the matrices $V_{t_1}, \dots, V_{t_n}$ are isometric, we have

$$U_t^* U_t = \begin{pmatrix} V_{t_1}^* & & \\ & \ddots & \\ & & V_{t_n}^* \end{pmatrix} \begin{pmatrix} V_{t_1} & & \\ & \ddots & \\ & & V_{t_n} \end{pmatrix} = \begin{pmatrix} V_{t_1}^* V_{t_1} & & \\ & \ddots & \\ & & V_{t_n}^* V_{t_n} \end{pmatrix} = I,$$

i.e., U_t is also isometric. Since we are looking for an isometric basis matrix V_t, we require

$$I = V_t^* V_t = (U_t \widehat{V}_t)^* (U_t \widehat{V}_t) = \widehat{V}_t^* U_t^* U_t \widehat{V}_t = \widehat{V}_t^* \widehat{V}_t,$$

i.e., we have to find an isometric matrix $\widehat{V}_t$.

Since U_t is isometric, $U_t U_t^*$ is an orthogonal projection. Let $z \in \mathbb{K}^{\mathcal{F}_t}$. The equations (7.1), applied to $P := U_t U_t^*$, $x := G_t z$, and $y := U_t \widehat{V}_t \widehat{V}_t^* U_t^*$, together with (4.4b) yield

$$\begin{aligned} \|(G_t - V_t V_t^* G_t) z\|_2^2 &= \|G_t z - U_t \widehat{V}_t \widehat{V}_t^* U_t^* z\|_2^2 \\ &= \|G_t z - U_t U_t^* G_t z\|_2^2 + \|U_t U_t^* G_t z - U_t \widehat{V}_t \widehat{V}_t^* U_t^* G_t z\|_2^2 \\ &= \|(G_t - U_t U_t^* G_t) z\|_2^2 + \|U_t (U_t^* G_t - \widehat{V}_t \widehat{V}_t^* U_t^* G_t) z\|_2^2 \\ &= \|(G_t - U_t U_t^* G_t) z\|_2^2 + \|(U_t^* G_t - \widehat{V}_t \widehat{V}_t^* U_t^* G_t) z\|_2^2. \end{aligned}$$

Introducing the matrix

$$\widehat{G}_t := U_t^* G_t \in \mathbb{K}^{m_t \times \mathcal{J}},$$

we obtain

$$\|(G_t - V_t V_t^* G_t) z\|_2^2 = \|(G_t - U_t U_t^* G_t) z\|_2^2 + \|(\widehat{G}_t - \widehat{V}_t \widehat{V}_t^* \widehat{G}_t) z\|_2^2. \tag{7.8}$$

The first term on the right-hand side takes the form

$$(G_t - U_t U_t^* G_t) z = \begin{pmatrix} (G|_{\hat{t}_1 \times \mathcal{F}_t} - V_{t_1} V_{t_1}^* G|_{\hat{t}_1 \times \mathcal{F}_t}) z \\ \vdots \\ (G|_{\hat{t}_n \times \mathcal{F}_t} - V_{t_n} V_{t_n}^* G|_{\hat{t}_n \times \mathcal{F}_t}) z \end{pmatrix},$$

i.e, it represents the error introduced in the children of t. Since we do not intend to change the basis matrices corresponding to the children, we consider this term fixed.

The second term on the right-hand side describes the approximation error introduced by the orthogonal projection $\widehat{V}_t \widehat{V}_t^*$, and this expression closely resembles the error term (7.6) for the leaf clusters. We can treat it by the same approach, i.e., by computing the singular value decomposition

$$\widehat{G}_t = \widehat{Q} \widehat{\Sigma} \widehat{P}^*$$

and using the first k_t columns of $\widehat{Q}$ to construct $\widehat{V}_t$. The basis matrix is now given by $V_t = U_t \widehat{V}_t$, and due to (7.7), the transfer matrices $E_{t_1}, \dots, E_{t_n}$ can be obtained by splitting $\widehat{V}_t$ into submatrices.

Efficient computation of $\widehat{G}_t$. Our algorithm requires the matrices

$$\widehat{G}_t = U_t^* G_t = \begin{pmatrix} V_{t_1}^* & & \\ & \ddots & \\ & & V_{t_n}^* \end{pmatrix} \begin{pmatrix} G_t|_{\hat{t}_1 \times \mathcal{F}_t} \\ \vdots \\ G_t|_{\hat{t}_n \times \mathcal{F}_t} \end{pmatrix} = \begin{pmatrix} V_{t_1}^* G_t|_{\hat{t}_1 \times \mathcal{F}_t} \\ \vdots \\ V_{t_n}^* G_t|_{\hat{t}_n \times \mathcal{F}_t} \end{pmatrix}$$

for all non-leaf clusters t. Computing these matrices directly is unattractive, even if we use the forward transformation to evaluate the products.

We can avoid the direct computation by introducing auxiliary matrices: if we let

$$X_t := V_t^* G_t \in \mathbb{K}^{k_t \times \mathcal{F}_t} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}},$$

we have

$$\widehat{G}_t = \begin{pmatrix} X_{t_1}|_{k_{t_1} \times \mathcal{F}_t} \\ \vdots \\ X_{t_n}|_{k_{t_n} \times \mathcal{F}_t} \end{pmatrix}, \tag{7.9}$$

i.e., we can obtain $\widehat{G}_t$ by copying those columns of $X_{t_1}, \ldots, X_{t_n}$ that are elements of $\mathcal{F}_t \subseteq \mathcal{F}_{t_1}, \ldots, \mathcal{F}_{t_n}$.

Of course, we require an efficient approach to construct these matrices. If $t \in \mathcal{T}_{\mathcal{I}}$ is a leaf, we can simply apply the definition to obtain X_t.

Otherwise, we have

$$X_t = V_t^* G_t = \widehat{V}_t^* U_t^* G_t = \widehat{V}_t^* \widehat{G}_t,$$

i.e., we can compute X_t using only the transfer matrices and $\widehat{G}_t$. The resulting algorithm is given in Figure 7.1.

Lemma 7.6 (Complexity) *The function `buildrowbasis_amatrix`, called with $t \in \mathcal{T}_{\mathcal{I}}$, requires not more than*

$$W_{bb}(t) := \begin{cases} C_{bb}\hat{k}|\mathcal{J}|\,|\hat{t}| & \text{if } \operatorname{chil}(t) = \emptyset, \\ \sum_{t' \in \operatorname{chil}(t)} C_{bb}\hat{k}|\mathcal{J}|k_{t'} + W_{bb}(t') & \text{otherwise} \end{cases} \qquad \text{operations,}$$

where $C_{bb} := \max\{1, C_{sn}\}C_{svd} + 2$ and

$$\hat{k} := \max\{r_{\mathcal{I}}, k_t \ :\ t \in \mathcal{T}_{\mathcal{I}}\}, \qquad C_{sn} := \max\{|\operatorname{chil}(t)| \ :\ t \in \mathcal{T}_{\mathcal{I}}\}$$

denote the maximal rank and the maximal number of children. We have

$$W_{bb}(t) \le C_{bb}\hat{k}|\mathcal{I}|\,|\mathcal{J}| + C_{bb}\hat{k}^2|\mathcal{T}_{\mathcal{I}}|\,|\mathcal{J}| \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}}.$$

Proof. We use induction over $|\mathcal{T}_t|$ to prove the complexity bound.

Let $t \in \mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t| = 1$. Then we have $\operatorname{chil}(t) = \emptyset$. The function first computes the SVD of G_t, this requires not more than

$$C_{\text{svd}}|\hat{t}|^2|\mathcal{F}_t| \le C_{\text{svd}}\hat{k}|\mathcal{J}|\,|\hat{t}| \quad \text{operations}$$

```
procedure buildrowbasis_amatrix($t$, $G$, var $V$, $X$);
  if $\operatorname{chil}(t) = \emptyset$ then begin
    Compute SVD of $G_t$;
    Choose rank $k_t$;
    Use first $k_t$ left singular vectors to form $V_t$;
    $X_t \leftarrow V_t^* G_t$
  end
  else begin
    for $t' \in \operatorname{chil}(t)$ do
      buildrowbasis_amatrix($t'$, $G$, $V$, $X$);
    Form $\widehat{G}_t$ according to (7.9);
    Compute SVD of $\widehat{G}_t$;
    Choose rank $k_t$;
    Use first $k_t$ left singular vectors to form $\widehat{V}_t$;
    $X_t \leftarrow \widehat{V}_t^* \widehat{G}_t$;
    Obtain transfer matrices by splitting $\widehat{V}_t$ according to (7.7)
  end
end
```

Figure 7.1: Adaptive row basis for a matrix G in array representation.

due to Assumption 5.7. Preparing X_t requires not more than

$$2k_t|\hat{t}|\,|\mathcal{F}_t| \leq 2\hat{k}|\mathcal{J}|\,|\hat{t}| \quad \text{operations,}$$

and we arrive at the upper bound

$$C_{\mathrm{svd}}\hat{k}|\mathcal{J}|\,|\hat{t}| + 2\hat{k}|\mathcal{J}|\,|\hat{t}| \leq W_{\mathrm{bb}}(t).$$

Now we assume that $n \in \mathbb{N}$ is given such that the number of operations can be bounded by $W_{\mathrm{bb}}(t)$ for all $t \in \mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t| \leq n$.

Let $t \in \mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t| = n+1$. For every child $t' \in \operatorname{chil}(t)$, we have $|\mathcal{T}_{t'}| \leq n$ and can apply the induction assumption. Once the bases for all children have been computed, we can form $\widehat{G}_t$ by copying the appropriate columns from $X_{t'}$, $t' \in \operatorname{chil}(t)$.

Since $\widehat{G}_t$ has $\sum_{t' \in \operatorname{chil}(t)} k_{t'}$ rows, Assumption 5.7 yields that the SVD requires not more than

$$C_{\mathrm{svd}} m_t^2 |\mathcal{F}_t| \leq C_{\mathrm{svd}} C_{\mathrm{sn}} \hat{k} |\mathcal{J}| \sum_{t' \in \operatorname{chil}(t)} k_{t'} \quad \text{operations.}$$

Computing X_t takes not more than

$$2k_t m_t |\mathcal{F}_t| \leq 2\hat{k}|\mathcal{J}| \sum_{t' \in \operatorname{chil}(t)} k_{t'} \quad \text{operations,}$$

and this leads to the upper bound

$$C_{\mathrm{sn}} C_{\mathrm{svd}} \hat{k} |\mathcal{J}| \sum_{t' \in \mathrm{chil}(t)} k_{t'} + 2\hat{k} |\mathcal{J}| \sum_{t' \in \mathrm{chil}(t)} k_{t'} + \sum_{t' \in \mathrm{chil}(t)} W_{\mathrm{bb}}(t') \le W_{\mathrm{bb}}(t).$$

In order to obtain the bound for the total complexity, we start with a straightforward induction to get

$$W_{\mathrm{bb}}(t) \le C_{\mathrm{bb}} \hat{k} |\mathcal{J}| \sum_{\substack{t \in \mathcal{T}_{\mathcal{I}} \\ \mathrm{chil}(t) = \emptyset}} |\hat{t}| + C_{\mathrm{bb}} \hat{k} |\mathcal{J}| \sum_{t \in \mathcal{T}_{\mathcal{I}}} \sum_{t' \in \mathrm{chil}(t)} k_{t'} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}}. \tag{7.10}$$

With Corollary 3.19 we obtain

$$\sum_{\substack{t \in \mathcal{T}_{\mathcal{I}} \\ \mathrm{chil}(t) = \emptyset}} |\hat{t}| = \sum_{t \in \mathcal{L}_{\mathcal{I}}} |\hat{t}| = \Big| \bigcup_{t \in \mathcal{L}_{\mathcal{I}}} \hat{t} \Big| = |\mathcal{I}|,$$

and since each cluster can have at most one parent we also have

$$\sum_{t \in \mathcal{T}_{\mathcal{I}}} \sum_{t' \in \mathrm{chil}(t)} k_{t'} = \sum_{t' \in \mathcal{T}_{\mathcal{I}}} \sum_{\substack{t \in \mathcal{T}_{\mathcal{I}} \\ t' \in \mathrm{chil}(t)}} k_{t'} \le \sum_{t' \in \mathcal{T}_{\mathcal{I}}} k_{t'} \le \hat{k} |\mathcal{T}_{\mathcal{I}}|.$$

Combining these estimates with (7.10) yields the required upper bound. ■

Remark 7.7 (Column basis) *Given the block tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ for a matrix $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$, we can define the block tree $\mathcal{T}^*_{\mathcal{I}\times\mathcal{J}}$ via*

$$b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}} \iff b^* := (s,t) \in \mathcal{T}^*_{\mathcal{I}\times\mathcal{J}} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}},\ s \in \mathcal{T}_{\mathcal{J}}.$$

If we apply `buildrowbasis_amatrix` *to G^* using the "adjoint" block tree $\mathcal{T}^*_{\mathcal{I}\times\mathcal{J}}$ for the row cluster tree $\mathcal{T}_{\mathcal{J}}$ and the column cluster tree $\mathcal{T}_{\mathcal{I}}$, we obtain a row cluster basis for G^* that due to (7.5) is a suitable column cluster basis for G.*

Remark 7.8 (Destructive compression) *Since the rank of G_t is always bounded by $|\hat{t}|$, we can overwrite the parts of G containing G_t by X_t once the matrix V_t is at our disposal.*

Following this approach, we only need a small amount of auxiliary storage for the matrices $\widehat{G}_t$ that can be discarded as soon as $\widehat{V}_t$ and X_t have been computed.

After `buildrowbasis_amatrix` *is complete, we have the matrices $V_t^* G|_{\hat{t}\times\hat{s}}$ at our disposal for all $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. If we apply the same procedure to the construction of the column basis, we can replace the submatrix $G|_{\hat{t}\times\hat{s}}$, which is no longer available, by $V_t^* G|_{\hat{t}\times\hat{s}}$ in the construction of $\widehat{G}_s$. This approach can reduce the complexity of this second step significantly, since $\widehat{G}_s$ now only has not more than $C_{\mathrm{sp}} \hat{k}(p_{\mathcal{I}} + 1)$ columns instead of $|\mathcal{I}|$.*

After the column basis is complete, we have the matrices $V_t^* G|_{\hat{t}\times\hat{s}} W_s$ at our disposal, i.e., the ideal coupling matrices for the $\mathcal{H}^2$-matrix.

7.3 Compression error

In our compression algorithm, every generation of descendants of a cluster introduces further approximation errors, and we have to investigate techniques for keeping these errors under control.

Since different clusters with different index sets can contribute to the same error, we introduce the matrices $\chi_t \in \mathbb{K}^{\mathcal{I}\times\hat{t}}$ for all clusters $t \in \mathcal{T}_{\mathcal{I}}$ via

$$\chi_{t,ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i \in \mathcal{I},\ j \in \hat{t}$$

that pad a vector $x \in \mathbb{K}^{\hat{t}}$ with zeros to obtain a vector $\chi_t x \in \mathbb{K}^{\mathcal{I}}$.

Let $t \in \mathcal{T}_{\mathcal{I}}$ with $n := |\operatorname{chil}(t)| > 0$ and $\operatorname{chil}(t) = \{t_1, \ldots, t_n\}$. Since the index sets of the children $t' \in \operatorname{chil}(t)$ form a disjoint partition of $\hat{t}$, we can write the equation

$$V_t|_{\hat{t}'} = V_{t'} E_{t'} \qquad \text{for all } t' \in \operatorname{chil}(t)$$

equivalently in the form

$$\chi_t V_t = \sum_{t' \in \operatorname{chil}(t)} \chi_{t'} V_{t'} E_{t'}.$$

We have

$$\begin{aligned} G_t - V_t V_t^* G_t &= G_t - U_t U_t^* G_t + U_t U_t^* G_t - V_t V_t^* G_t \\ &= \begin{pmatrix} G_t|_{\hat{t}_1\times\mathcal{F}_t} - V_{t_1} V_{t_1}^* G_t|_{\hat{t}_1\times\mathcal{F}_t} \\ \vdots \\ G_t|_{\hat{t}_n\times\mathcal{F}_t} - V_{t_n} V_{t_n}^* G_t|_{\hat{t}_n\times\mathcal{F}_t} \end{pmatrix} + U_t U_t^* G_t - U_t \widehat{V}_t \widehat{V}_t^* U_t^* G_t \\ &= \begin{pmatrix} G_{t_1}|_{\hat{t}_1\times\mathcal{F}_t} - V_{t_1} V_{t_1}^* G_{t_1}|_{\hat{t}_1\times\mathcal{F}_t} \\ \vdots \\ G_{t_n}|_{\hat{t}_n\times\mathcal{F}_t} - V_{t_n} V_{t_n}^* G_{t_n}|_{\hat{t}_n\times\mathcal{F}_t} \end{pmatrix} + U_t(\widehat{G}_t - \widehat{V}_t \widehat{V}_t^* \widehat{G}_t). \end{aligned}$$

Using the "padding matrices" χ_t, this equation can be written in the form

$$\begin{aligned} \chi_t(G_t - V_t V_t^* G_t) = \chi_t U_t(\widehat{G}_t - \widehat{V}_t \widehat{V}_t^* \widehat{G}_t) \\ + \sum_{t' \in \operatorname{chil}(t)} \chi_{t'}(G_{t'} - V_{t'} V_{t'}^* G_{t'})|_{\hat{t}'\times\mathcal{F}_t}. \end{aligned} \tag{7.11}$$

The first term on the right of this equation is under our control: U_t is isometric, i.e., it will not influence the norm, and the approximation error of $\widehat{G}_t$ can be controlled via choosing an appropriate rank following the singular value decomposition. The remaining terms have a similar form as the original error on the left, only for the children t' instead of t. This property suggests that we apply (7.11) recursively to the children until we arrive at leaves, where we can control the error explicitly.

Lemma 7.9 (Error sum) *We define*

$$D_t := \begin{cases} G_t - V_t V_t^* G_t & \text{if } \operatorname{chil}(t) = \emptyset, \\ U_t(\widehat{G}_t - \widehat{V}_t \widehat{V}_t^* \widehat{G}_t) & \text{otherwise}, \end{cases} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}}. \tag{7.12}$$

We have

$$\chi_t (G_t - V_t V_t^* G_t) = \sum_{t^* \in \mathcal{T}_t} \chi_{t^*} D_{t^*}|_{\hat{t}^* \times \mathcal{F}_t} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}}.$$

Proof. By induction over $|\mathcal{T}_t|$.

Let $t \in \mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t| = 1$. This implies $\operatorname{chil}(t) = \emptyset$ and the equation follows directly from the definition of D_t.

Let now $n \in \mathbb{N}$ be given such that the equation holds for all $t \in \mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t| \leq n$.

Let $t \in \mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t| = n + 1$. Then we have $\operatorname{chil}(t) \neq \emptyset$ and can apply (7.11) to obtain

$$\begin{aligned} \chi_t (G_t - V_t V_t^* G_t) &= \chi_t D_t + \sum_{t' \in \operatorname{chil}(t)} \chi_{t'} (G_t|_{\hat{t}' \times \mathcal{F}_t} - V_{t'} V_{t'}^* G_t|_{\hat{t}' \times \mathcal{F}_t}) \\ &= \chi_t D_t + \sum_{t' \in \operatorname{chil}(t)} \chi_{t'} (G_{t'}|_{\hat{t}' \times \mathcal{F}_t} - V_{t'} V_{t'}^* G_{t'}|_{\hat{t}' \times \mathcal{F}_t}) \\ &= \chi_t D_t + \sum_{t' \in \operatorname{chil}(t)} \chi_{t'} (G_{t'} - V_{t'} V_{t'}^* G_{t'})|_{\hat{t}' \times \mathcal{F}_t} \end{aligned}$$

due to $\mathcal{F}_t \subseteq \mathcal{F}_{t'}$ for all $t' \in \operatorname{chil}(t)$.

Since $|\mathcal{T}_{t'}| \leq n$ holds for all $t' \in \operatorname{chil}(t)$, we can use the induction assumption to get

$$\begin{aligned} \chi_t (G_t - V_t V_t^* G_t) &= \chi_t D_t + \sum_{t' \in \operatorname{chil}(t)} \chi_{t'} (G_{t'} - V_{t'} V_{t'}^* G_{t'})|_{\hat{t}' \times \mathcal{F}_t} \\ &= \chi_t D_t + \sum_{t' \in \operatorname{chil}(t)} \chi_{t'} \Big(\sum_{t^* \in \mathcal{T}_{t'}} \chi_{t^*} D_{t^*}|_{\hat{t}^* \times \mathcal{F}_t} \Big)|_{\hat{t}' \times \mathcal{F}_t} \\ &= \chi_t D_t + \sum_{t' \in \operatorname{chil}(t)} \sum_{t^* \in \mathcal{T}_{t'}} \chi_{t^*} D_{t^*}|_{\hat{t}^* \times \mathcal{F}_t} \\ &= \sum_{t^* \in \mathcal{T}_t} \chi_{t^*} D_{t^*}|_{\hat{t}^* \times \mathcal{F}_t}. \end{aligned}$$

This completes the induction step. ∎

Lemma 7.9 allows us to split the total error $G_t - V_t V_t^* G_t$ for the approximation of G_t into the contributions $\widehat{G}_{t^*} - \widehat{V}_{t^*} \widehat{V}_{t^*}^* \widehat{G}_{t^*}$ and $G_{t^*} - V_{t^*} V_{t^*}^* G_{t^*}$ of its decendants, all of which we can control explicitly via the singular value decomposition.

We could now obtain an estimate for the error by simply applying the triangle inequality, but this would not provide a sharp estimate. A closer look reveals that the ranges of the matrices $\chi_{t^*} D_{t^*}$ are, in fact, orthogonal, so we can use Pythagoras' identity to obtain an error equation instead of an error estimate.

Lemma 7.10 (Error orthogonality) *Let $t \in \mathcal{T}_{\mathcal{I}}$ and $t_1, t_2 \in \mathcal{T}_t$ with $t_1 \neq t_2$. For the matrices defined in (7.12), we have*

$$\langle \chi_{t_1} D_{t_1} x|_{\mathcal{F}_{t_1}}, \chi_{t_2} D_{t_2} y|_{\mathcal{F}_{t_2}} \rangle = 0 \qquad \text{for all } x, y \in \mathbb{K}^{\mathcal{J}}.$$

Proof. Let $x, y \in \mathbb{K}^{\mathcal{J}}$. If $\hat{t}_1 \cap \hat{t}_2 = \emptyset$, we have $\chi_{t_1}^* \chi_{t_2} = 0$ and therefore

$$\langle \chi_{t_1} D_{t_1} x|_{\mathcal{F}_{t_1}}, \chi_{t_2} D_{t_2} y|_{\mathcal{F}_{t_2}} \rangle = \langle D_{t_1} x|_{\mathcal{F}_{t_1}}, \chi_{t_1}^* \chi_{t_2} D_{t_2} y|_{\mathcal{F}_{t_2}} \rangle = 0.$$

Otherwise, we assume level$(t_2) \geq$ level(t_1) without loss of generality and apply Lemma 3.18 to obtain $t_2 \in \operatorname{desc}(t_1)$. Due to $t_1 \neq t_2$, we can find $t' \in \operatorname{chil}(t_1)$ such that $t_2 \in \mathcal{T}_{t'}$. We have $D_{t_1} x|_{\mathcal{F}_{t_1}} \in \operatorname{range}(U_{t_1})$ by construction and therefore also $(D_{t_1} x|_{\mathcal{F}_{t_1}})|_{\hat{t}'} \in \operatorname{range}(V_{t'})$, i.e., we can find $z \in \mathbb{K}^{k_{t'}}$ with

$$(D_{t_1} x|_{\mathcal{F}_{t_1}})|_{\hat{t}'} = V_{t'} z.$$

Since $t_2 \in \mathcal{T}_{t'}$, we can use Lemma 6.14 to obtain

$$(D_{t_1} x|_{\mathcal{F}_{t_1}})|_{\hat{t}_2} = (V_{t'} z)|_{\hat{t}_2} = V_{t_2} E_{t_2,t'} z$$

and use $\hat{t}_2 \subseteq \hat{t}_1$ to arrive at

$$\begin{aligned}
\langle \chi_{t_1} D_{t_1} x|_{\mathcal{F}_{t_1}}, \chi_{t_2} D_{t_2} y|_{\mathcal{F}_{t_2}} \rangle &= \langle \chi_{t_2}^* \chi_{t_1} D_{t_1} x|_{\mathcal{F}_{t_1}}, D_{t_2} y|_{\mathcal{F}_{t_2}} \rangle \\
&= \langle (D_{t_1} x|_{\mathcal{F}_{t_1}})|_{\hat{t}_2}, D_{t_2} y|_{\mathcal{F}_{t_2}} \rangle \\
&= \langle V_{t_2} E_{t_2,t'} z, D_{t_2} y|_{\mathcal{F}_{t_2}} \rangle.
\end{aligned}$$

If $\operatorname{chil}(t_2) = \emptyset$, we have

$$\begin{aligned}
\langle \chi_{t_1} D_{t_1} x|_{\mathcal{F}_{t_1}}, \chi_{t_2} D_{t_2} y|_{\mathcal{F}_{t_2}} \rangle &= \langle V_{t_2} E_{t_2,t'} z, D_{t_2} y|_{\mathcal{F}_{t_2}} \rangle \\
&= \langle V_{t_2} E_{t_2,t'} z, (G_{t_2} - V_{t_2} V_{t_2}^* G_{t_2}) y|_{\mathcal{F}_{t_2}} \rangle \\
&= \langle E_{t_2,t'} z, V_{t_2}^* (G_{t_2} - V_{t_2} V_{t_2}^* G_{t_2}) y|_{\mathcal{F}_{t_2}} \rangle = 0
\end{aligned}$$

due to $V_{t_2}^* V_{t_2} = I$. Otherwise, i.e., if $\operatorname{chil}(t_2) \neq \emptyset$ holds, we find

$$\begin{aligned}
\langle \chi_{t_1} D_{t_1} x|_{\mathcal{F}_{t_1}}, \chi_{t_2} D_{t_2} y|_{\mathcal{F}_{t_2}} \rangle &= \langle V_{t_2} E_{t_2,t'} z, D_{t_2} y|_{\mathcal{F}_{t_2}} \rangle \\
&= \langle U_{t_2} \widehat{V}_{t_2} E_{t_2,t'} z, U_{t_2} (\widehat{G}_{t_2} - \widehat{V}_{t_2} \widehat{V}_{t_2}^* \widehat{G}_{t_2}) y|_{\mathcal{F}_{t_2}} \rangle \\
&= \langle E_{t_2,t'} z, \widehat{V}_{t_2}^* U_{t_2}^* U_{t_2} (\widehat{G}_{t_2} - \widehat{V}_{t_2} \widehat{V}_{t_2}^* \widehat{G}_{t_2}) y|_{\mathcal{F}_{t_2}} \rangle \\
&= \langle E_{t_2,t'} z, \widehat{V}_{t_2}^* (\widehat{G}_{t_2} - \widehat{V}_{t_2} \widehat{V}_{t_2}^* \widehat{G}_{t_2}) y|_{\mathcal{F}_{t_2}} \rangle = 0
\end{aligned}$$

due to $U_{t_2}^* U_{t_2} = I$ and $\widehat{V}_{t_2}^* \widehat{V}_{t_2} = I$. ∎

Lemma 7.9 allows us to represent the approximation error by a sum of the contributions introduced in all clusters, and Lemma 7.10 states that all of the contributions are orthogonal with respect to each other, so that we can use the Pythagoras identity to obtain an expression for the total error that involves only quantities that we can control explicitly.

Theorem 7.11 (Error decomposition) *We define the approximation $\widetilde{G} \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ by*

$$\widetilde{G}|_{\hat{t}\times\hat{s}} := \begin{cases} G|_{\hat{t}\times\hat{s}} & \text{if } b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}, \\ V_t V_t^* G|_{\hat{t}\times\hat{s}} & \text{otherwise} \end{cases} \qquad \text{for all } b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}.$$

Using the matrices defined in (7.12), we have

$$(G-\widetilde{G})z = \sum_{t\in\mathcal{T}_\mathcal{I}} \chi_t D_t z|_{\mathcal{F}_t} \qquad \text{for all } z\in\mathbb{K}^\mathcal{J},$$

$$\|(G-\widetilde{G})z\|_2^2 = \sum_{t\in\mathcal{T}_\mathcal{I}} \|D_t z|_{\mathcal{F}_t}\|_2^2 \qquad \text{for all } z\in\mathbb{K}^\mathcal{J},$$

$$\|G-\widetilde{G}\|_2^2 \le \sum_{t\in\mathcal{T}_\mathcal{I}} \|D_t\|_2^2.$$

Proof. Let $z\in\mathbb{K}^\mathcal{J}$. Corollary 3.23 yields

$$(G-\widetilde{G})z = \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{J}}} \chi_t (G-\widetilde{G})|_{\hat{t}\times\hat{s}} z|_{\hat{s}} = \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \chi_t (G_t - V_t V_t^* G_t)|_{\hat{t}\times\hat{s}} z|_{\hat{s}},$$

and we can use Lemma 7.9 to get

$$\chi_t (G_t - V_t V_t^* G_t)|_{\hat{t}\times\hat{s}} z|_{\hat{s}} = \sum_{t^*\in\mathcal{T}_t} \chi_{t^*} D_{t^*}|_{\hat{t}^*\times\hat{s}} z|_{\hat{s}} \qquad \text{for all } t\in\mathcal{T}_\mathcal{I}.$$

Combining both equations and recalling the definition of $\mathcal{F}_t$ of Lemma 6.15 gives us

$$\begin{aligned}(G-\widetilde{G})z &= \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} \sum_{t^*\in\mathcal{T}_t} \chi_{t^*} D_{t^*}|_{\hat{t}^*\times\hat{s}} z|_{\hat{s}} = \sum_{t\in\mathcal{T}_\mathcal{I}} \sum_{s\in\operatorname{row}^+(t)} \sum_{t^*\in\mathcal{T}_t} \chi_{t^*} D_{t^*}|_{\hat{t}^*\times\hat{s}} z|_{\hat{s}} \\ &= \sum_{t^*\in\mathcal{T}_\mathcal{I}} \sum_{t\in\operatorname{pred}(t^*)} \sum_{s\in\operatorname{row}^+(t)} \chi_{t^*} D_{t^*}|_{\hat{t}^*\times\hat{s}} z|_{\hat{s}} = \sum_{t^*\in\mathcal{T}_\mathcal{I}} \chi_{t^*} D_{t^*} z|_{\mathcal{F}_{t^*}}.\end{aligned}$$

This is the first equation. Lemma 7.10 guarantees that all of its terms are pairwise orthogonal, and this allows us to obtain a version of the Pythagoras identity for the error decomposition.

$$\begin{aligned}\|(G-\widetilde{G})z\|_2^2 &= \langle (G-\widetilde{G})z, (G-\widetilde{G})z\rangle = \Big\langle \sum_{t_1\in\mathcal{T}_\mathcal{I}} \chi_{t_1} D_{t_1} z|_{\mathcal{F}_{t_1}}, \sum_{t_2\in\mathcal{T}_\mathcal{I}} \chi_{t_2} D_{t_2} z|_{\mathcal{F}_{t_2}} \Big\rangle \\ &= \sum_{t_1\in\mathcal{T}_\mathcal{I}} \sum_{t_2\in\mathcal{T}_\mathcal{I}} \langle \chi_{t_1} D_{t_1} z|_{\mathcal{F}_{t_1}}, \chi_{t_2} D_{t_2} z|_{\mathcal{F}_{t_2}} \rangle \\ &= \sum_{t\in\mathcal{T}_\mathcal{I}} \langle \chi_t D_t z|_{\mathcal{F}_t}, \chi_t D_t z|_{\mathcal{F}_t}\rangle = \sum_{t\in\mathcal{T}_\mathcal{I}} \|\chi_t D_t z|_{\mathcal{F}_t}\|^2 = \sum_{t\in\mathcal{T}_\mathcal{I}} \|D_t z|_{\mathcal{F}_t}\|^2.\end{aligned}$$

This is the second equation. It gives rise to the bound for the spectral norm if we take the maximum on both sides. ■

Theorem 7.11 provides us with an equation and an estimate for the total error of the approximation. Frequently, we are interested in blockwise estimates, e.g., if we want to perform arithmetic operations based on block decompositions as in Chapter 5.

Corollary 7.12 (Block error) *Let $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. We have*

$$\|(G|_{\hat{t}\times\hat{s}} - V_t V_t^* G|_{\hat{t}\times\hat{s}})z\|^2 = \sum_{t^*\in\mathcal{T}_t} \|D_{t^*}|_{\hat{t}^*\times\hat{s}} z\|^2 \qquad \text{for all } z \in \mathbb{K}^{\hat{s}}.$$

Proof. Let $z \in \mathbb{K}^{\hat{s}}$. Due to $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, we have $\hat{s} \subseteq \mathcal{F}_t$. We let $x := \chi_s z \in \mathbb{K}^{\mathcal{J}}$ and use Lemma 7.9 and Lemma 7.10 to obtain

$$\begin{aligned}
\|(G|_{\hat{t}\times\hat{s}} - V_t V_t^* G|_{\hat{t}\times\hat{s}})z\|^2 &= \|(G_t - V_t V_t^* G_t)x\|^2 = \Big\| \sum_{t^*\in\mathcal{T}_t} \chi_{t^*} D_{t^*} x|_{\mathcal{F}_{t^*}} \Big\|^2 \\
&= \sum_{t_1\in\mathcal{T}_t} \sum_{t_2\in\mathcal{T}_t} \langle \chi_{t_1} D_{t_1} x|_{\mathcal{F}_{t_1}}, \chi_{t_2} D_{t_2} x|_{\mathcal{F}_{t_2}} \rangle \\
&= \sum_{t^*\in\mathcal{T}_t} \|\chi_{t^*} D_{t^*} x|_{\mathcal{F}_{t^*}}\|^2 = \sum_{t^*\in\mathcal{T}_t} \|D_{t^*}|_{\hat{t}^*\times\hat{s}} z\|^2.
\end{aligned}$$

In the final step, we have used $x_j = 0$ for all $j \in \mathcal{J} \setminus \hat{s}$ and $x_j = z_j$ for all $j \in \hat{s}$. ■

Controlling the error matrices D_t is straightforward: for leaves $t \in \mathcal{L}_{\mathcal{I}}$, we have

$$\|D_t\|_2 = \|G_t - V_t V_t^* G_t\|_2,$$

and this is precisely the approximation error that we can control by using the left singular vectors of G_t to define V_t. For the non-leaf clusters $t \in \mathcal{T}_{\mathcal{I}} \setminus \mathcal{L}_{\mathcal{I}}$, we have

$$\|D_t\|_2 = \|U_t(\widehat{G}_t - \widehat{V}_t \widehat{V}_t^* \widehat{G}_t)\|_2 = \|\widehat{G}_t - \widehat{V}_t \widehat{V}_t^* \widehat{G}_t\|_2$$

since U_t is isometric, and again this is the approximation error that we can control by using the left singular vectors of $\widehat{G}_t$ to obtain the transfer matrices $\widehat{V}_t$.

Remark 7.13 (Weighted estimates) *The compression error can be controlled in far more detail by introducing weight factors to the matrices G_t: if we have $\omega_{t,s} \in \mathbb{R}_{>0}$ for all $t \in \mathcal{T}_{\mathcal{I}}$ and $s \in \operatorname{row}(t^+)$ for $t^+ \in \operatorname{pred}(t)$, we can define $G_{t,\omega} \in \mathbb{K}^{\hat{t}\times\mathcal{F}_t}$ by*

$$G_{t,\omega}|_{\hat{t}\times\hat{s}} := \omega_{t,s}^{-1} G|_{\hat{t}\times\hat{s}} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}},\ t^+ \in \operatorname{pred}(t),\ s \in \operatorname{row}(t^+)$$

and replace G_t by $G_{t,\omega}$ in the SVD and accordingly D_t by $D_{t,\omega}$. If we now choose the ranks k_t to ensure

$$\|D_{t,\omega}\|_2 \leq \epsilon \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}},$$

Corollary 7.12 yields the estimate

$$\|G|_{\hat{t}\times\hat{s}} - V_t V_t^* G|_{\hat{t}\times\hat{s}}\|_2^2 \leq \sum_{t^*\in\mathcal{T}_t} \omega_{t^*,s}^2 \epsilon^2 \qquad \text{for all } b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},$$

i.e., we can ensure different error bounds for different blocks [5].

One possibility are block-relative error estimates: if we choose the weights $\omega_{t,s} := \|G|_{\hat{t}^+\times\hat{s}}\|_2\, q^{\operatorname{level}(t)-\operatorname{level}(t^+)}$ for $t \in \mathcal{T}_{\mathcal{I}}$, $t^+ \in \operatorname{pred}(t)$ and $s \in \operatorname{row}(t^+)$, our approach yields

$$\|G|_{\hat{t}\times\hat{s}} - V_t V_t^* G|_{\hat{t}\times\hat{s}}\|_2^2 \leq \frac{\epsilon}{1 - C_{sn} q^2} \|G|_{\hat{t}\times\hat{s}}\|_2^2 \qquad \text{for all } b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},$$

where C_{sn} is a bound for the number of sons and $C_{sn} q^2 < 1$.

7.4 Compression of $\mathcal{H}$-matrices

The algorithm `buildrowbasis_amatrix` requires at least $|\mathcal{I}|\,|\mathcal{J}|$ operations. This is not surprising, since it is looking for an approximation of a *general* matrix $G \in \mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ and therefore has to examine each of its coefficients at least once.

If the matrix G is already given in compressed form, e.g., as an $\mathcal{H}$- or $\mathcal{H}^2$-matrix, we can take advantage of the additional structure to significantly reduce the computational work required for the compression.

Let us assume that G is an $\mathcal{H}$-matrix, i.e., that we have

$$G|_{\hat{t}\times\hat{s}} = A_b B_b^* \qquad\qquad \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}.$$

In order to take advantage of this factorization, we introduce

$$\mathcal{B}_t := \{b = (t_\top, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}} \;:\; t_\top \in \operatorname{pred}(t)\} \qquad\qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}}$$

and have

$$\mathcal{F}_t = \bigcup_{(t_\top,s)\in\mathcal{B}_t} \hat{s}$$

due to (6.10).

Given a cluster $t \in \mathcal{T}_{\mathcal{I}}$, we let $m := |\mathcal{B}_t|$ and $\mathcal{B}_t = \{b_1 = (t_{\top,1}, s_1), \ldots, b_m = (t_{\top,m}, s_m)\}$ and write G_t in the form

$$G_t = \begin{pmatrix} G|_{\hat{t}\times\hat{s}_1} & \ldots & G|_{\hat{t}\times\hat{s}_m} \end{pmatrix}.$$

Since our construction requires only the singular values and the left singular vectors, we can replace G_t by an isometric factorization without losing the relevant information.

To this end, we compute QR factorizations

$$B_b = Q_b R_b, \qquad Q_b \in \mathbb{K}^{\hat{s}\times\mathcal{L}_b} \text{ isometric}, R_b \in \mathbb{K}^{\mathcal{L}_b\times k} \qquad \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},$$

where the index sets $\mathcal{L}_b$ are chosen as subsets of $\hat{s}$ with $|\mathcal{L}_b| \le k$. Due to Corollary 3.23, this choice implies that the sets $\mathcal{L}_{b_1} \subseteq \hat{s}_1, \ldots, \mathcal{L}_{s_m} \subseteq \hat{s}_m$ are pairwise disjoint.

The factorizations of the matrices B_b give rise to a factorization of G_t:

$$\begin{aligned} G_t &= \begin{pmatrix} G|_{\hat{t}\times\hat{s}_1} & \ldots & G|_{\hat{t}\times\hat{s}_m} \end{pmatrix} = \begin{pmatrix} A_{b_1}|_{\hat{t}\times k} R_{b_1}^* Q_{b_1}^* & \ldots & A_{b_m}|_{\hat{t}\times k} R_{b_m}^* Q_{b_m}^* \end{pmatrix} \\ &= \begin{pmatrix} A_{b_1}|_{\hat{t}\times k} R_{b_1}^* & \ldots & A_{b_m}|_{\hat{t}\times k} R_{b_m}^* \end{pmatrix} \begin{pmatrix} Q_{b_1}^* & & \\ & \ddots & \\ & & Q_{b_m}^* \end{pmatrix}. \end{aligned}$$

Since $Q_{(t,s_1)}, \ldots, Q_{(t,s_m)}$ are isometric, the second factor does not influence the singular values or left singular vectors, so we can discard it and conclude that we can replace G_t by the *condensed* matrix

$$G_t^c := \begin{pmatrix} A_{b_1}|_{\hat{t}\times k} R_{b_1}^* & \ldots & A_{b_m}|_{\hat{t}\times k} R_{b_m}^* \end{pmatrix} \in \mathbb{K}^{\hat{t}\times\mathcal{F}_t^c}, \qquad \mathcal{F}_t^c := \bigcup_{b\in\mathcal{B}_t} \mathcal{L}_b \subseteq \mathcal{J}.$$

In a similar fashion, we replace $X_t = V_t^* G_t$ by

$$X_t^c := V_t^* G_t^c$$

and for clusters t with $\operatorname{chil}(t) \neq \emptyset$, we let $n := |\operatorname{chil}(t)|$ and $\operatorname{chil}(t) = \{t_1, \ldots, t_n\}$ and replace $\widehat{G}_t$ by its condensed counterpart

$$\widehat{G}_t^c = \begin{pmatrix} X_{t_1}^c|_{k_{t_1} \times \mathcal{F}_t^c} \\ \vdots \\ X_{t_n}^c|_{k_{t_n} \times \mathcal{F}_t^c} \end{pmatrix}.$$

This approach reduces the computational work significantly: there are at most $p_{\mathcal{I}} + 1$ predecessors of a cluster $t \in \mathcal{T}_{\mathcal{I}}$, and each of these predecessors can be associated with at most C_{sp} blocks. The thin QR factorizations guarantee that in the condensed matrix, every one of these blocks is represented by at most k columns.

Lemma 7.14 (Condensed farfield) *We assume that $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is C_{sp}-sparse. Then we have*

$$|\mathcal{F}_t^c| \leq C_{sp} k \,|\operatorname{pred}(t)| \leq C_{sp} k (p_{\mathcal{I}} + 1) \qquad \textit{for all } t \in \mathcal{T}_{\mathcal{I}}$$

Proof. Let $t \in \mathcal{T}_{\mathcal{I}}$. We have

$$\mathcal{B}_t = \{b = (t_\top, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}} \ : \ t_\top \in \operatorname{pred}(t)\} \subseteq \bigcup_{t_\top \in \operatorname{pred}(t)} \operatorname{row}(t_\top),$$

$$|\mathcal{B}_t| \leq \sum_{t_\top \in \operatorname{pred}(t)} |\operatorname{row}(t_\top)| \leq C_{\mathrm{sp}} \,|\operatorname{pred}(t)| \leq C_{\mathrm{sp}}(p_{\mathcal{I}} + 1).$$

Due to

$$|\mathcal{L}_b| \leq k \qquad \text{for all } b \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},$$

we obtain

$$|\mathcal{F}_t^c| \leq \sum_{b \in \mathcal{B}_t} |\mathcal{L}_b| \leq k\,|\mathcal{B}_t| \leq C_{\mathrm{sp}} k \operatorname{pred}(t) \leq C_{\mathrm{sp}} k (p_{\mathcal{I}} + 1).$$

■

Lemma 7.15 (Complexity) *Preparing the weight matrices R_b for all $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ requires not more than*

$$C_{qr} C_{sp} k^2 (p_{\mathcal{I}\times\mathcal{J}} + 1)|\mathcal{J}| \quad \textit{operations.}$$

If we replace G_t, $\widehat{G}_t$ and X_t in the algorithm `buildrowbasis_amatrix` by their condensed counterparts G_t^c, $\widehat{G}_t^c$ and X_t^c, the resulting algorithm requires not more than

$$C_{sp} C_{bb} \hat{k}^2 (p_{\mathcal{I}} + 1)|\mathcal{I}| + C_{sp} C_{bb} \hat{k}^3 (p_{\mathcal{I}} + 1)|\mathcal{T}_{\mathcal{I}}| \quad \textit{operations.}$$

Proof. Let $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. Due to Assumption 5.7, the QR factorization of B_b can be computed in not more than

$$C_{\mathrm{qr}} k^2 \, |\hat{s}| \quad \text{operations,}$$

and performing this task for all admissible leaves requires not more than

$$C_{\mathrm{qr}} k^2 \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} |\hat{s}| \le C_{\mathrm{sp}} C_{\mathrm{qr}} k^2 (p_{\mathcal{I}\times\mathcal{J}} + 1)|\mathcal{J}| \quad \text{operations}$$

due to Lemma 3.34.

In order to obtain the estimate for the construction of the row basis, we simply replace $\mathcal{F}_t$ by $\mathcal{F}^c_t$ in the proof of Lemma 7.6 and use Lemma 7.14 to find a bound for $|\mathcal{F}^c_t|$. ■

Once we have constructed suitable isometric cluster bases $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $W = (W_s)_{s\in\mathcal{T}_\mathcal{J}}$, we can compute the coupling matrices for admissible leaves efficiently by taking advantage of the low-rank factorization: we have

$$S_b := V_t^* G|_{\hat{t}\times\hat{s}} W_s = V_t^* A_b B_b^* W_s = (V_t^* A_b)(W_s^* B_b)^* \qquad \text{for all } b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},$$

and $V_t^* A_b \in \mathbb{K}^{K_t\times k}$ and $W_s^* B_b \in \mathbb{K}^{k_s\times k}$ can be evaluated by applying the forward transformation to the columns of A_b and B_b, and this takes not more than

$$2k^2|\hat{t}| + 2k^3|\mathcal{T}_t| + 2k^2|\hat{s}| + 2k^3|\mathcal{T}_s| \quad \text{operations.}$$

Multiplying both products requires only $2\,k_t k k_s \le 2k^3$ operations per block.

Remark 7.16 (Typical complexity) *If we have $|\hat{t}| \ge k$ and $|\hat{s}| \ge k$ for all leaves $t \in \mathcal{L}_\mathcal{I}$ and $s \in \mathcal{L}_\mathcal{J}$ and if no cluster has exactly one child, we have already seen in Remark 6.10 that we have $k|\mathcal{T}_\mathcal{I}| \le 2|\mathcal{I}|$ and $k|\mathcal{T}_\mathcal{J}| \le 2|\mathcal{J}|$ and find that our algorithm requires $\mathcal{O}(\hat{k}^2(p_\mathcal{I} + 1)(|\mathcal{I}| + |\mathcal{J}|))$ operations. The original $\mathcal{H}$-matrix requires $\mathcal{O}(\hat{k}(p_\mathcal{I} + 1)(|\mathcal{I}| + |\mathcal{J}|))$ units of storage, so the algorithm has almost optimal complexity.*

Remark 7.17 (Error control) *Since this algorithm computes* exactly *the same cluster bases (up to rounding errors) as the previous one, the same error estimates apply.*

Remark 7.18 (Second phase) *Usually we have to compute both a row and a column basis. If we start with the row basis, we compute QR factorizations of the matrices B_b and the condensed matrices $A_b R_b^*$.*

The column basis can be computed by reversing the roles of A_b and B_b, i.e., the QR factorizations of A_b have to be computed and the triangular factors multiplied with B_b.

There is, however, a potentially attractive alternative: if we already have an isometric row basis at our disposal, we can also condense the admissible blocks by replacing A_b by $V_t V_t^* A_b$ and $B_b A_b^*$ by $B_b A_b^* V_t V_t^*$. Since V_t^* is isometric, we can discard it as before to obtain a condensed matrix G^c_t. This approach has the advantage that the matrices $\widehat{G}^c_t$ end up containing the coupling matrices for the compressed $\mathcal{H}^2$-matrix.

7.5 Compression of $\mathcal{H}^2$-matrices

It is frequently desirable to construct an $\mathcal{H}^2$-matrix approximation of a matrix that is already, explicitly or implicitly, given in $\mathcal{H}^2$-matrix form, e.g., to reduce an unnecessarily high rank or to recompress the intermediate result of an arithmetic operation.

We assume that G is an $\mathcal{H}^2$-matrix with a row cluster basis V_{old} and a column cluster basis W_{old}, both of rank k. The transfer matrices for both bases are denoted by $(E_{\mathrm{old},t})_{t\in\mathcal{T}_\mathcal{I}}$ and $(F_{\mathrm{old},s})_{s\in\mathcal{T}_\mathcal{J}}$, and the coupling matrices by $(S_{\mathrm{old},b})_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$.

In order to handle the recompression efficiently, we can take advantage of Corollary 6.17: for each $t\in\mathcal{T}_\mathcal{I}$, we know that a matrix $B_t\in\mathbb{K}^{\mathcal{F}_t\times k}$ exists satisfying

$$G_t = G|_{\hat{t}\times\mathcal{F}_t} = V_{\mathrm{old},t}B_t^*.$$

As in the case of the $\mathcal{H}$-matrix compression, we can use a QR factorization

$$B_t = Q_t Z_t,$$

where $Z_t\in\mathbb{K}^{\mathcal{L}_t\times k}$ for an index set $\mathcal{L}_t\subseteq\mathcal{F}_t$ with $|\mathcal{L}_t|\le k$ to find a representation

$$G_t = V_{\mathrm{old},t}Z_t^*Q_t^*$$

that allows us to discard the isometric matrix Q_t and only compute the SVD of the condensed matrix $V_{\mathrm{old},t}Z_t^*$. Since this matrix cannot have more than k columns, the resulting recompression algorithm is very efficient.

Definition 7.19 (Total weights) *A family $(Z_t)_{t\in\mathcal{T}_\mathcal{I}}$ of matrices $Z_t\in\mathbb{K}^{\mathcal{L}_t\times k}$ with index sets $\mathcal{L}_t\subseteq\mathcal{F}_t$ satisfying $|\mathcal{L}_t|\le k$ is called a family of* total weights *for the row cluster basis V_{old} and the matrix $G\in\mathbb{K}^{\mathcal{I}\times\mathcal{J}}$ if for every $t\in\mathcal{T}_\mathcal{I}$ there is an isometric matrix $Q_t\in\mathbb{K}^{\mathcal{F}_t\times\mathcal{L}_t}$ such that*

$$G_t = V_{old,t}Z_t^*Q_t^*. \tag{7.13}$$

Total weight matrices allow us to significantly reduce the computational work required for the $\mathcal{H}^2$-matrix compression, but constructing them directly via the definition (7.13) is unattractive: computing Z_t directly would require $\sim k^2\,|\mathcal{F}_t|$ operations and lead to quadratic complexity if applied to all clusters.

In order to overcome this obstacle, we follow the same approach as for the forward and backward transformation and consider the task of computing isometric matrices $Q_t\in\mathbb{K}^{\mathcal{F}_t\times\mathcal{L}_t}$ and weight matrices $Z_t\in\mathbb{K}^{\mathcal{L}_t\times k}$ with $|\mathcal{L}_t|\le k$ such that

$$G_t = V_{\mathrm{old},t}Z_t^*Q_t^* \qquad\qquad \text{for all } t\in\mathcal{T}_\mathcal{I}$$

for *all* clusters, not just for individual clusters. As in the transformation algorithms, this allows us to make use of the hierarchical structure to speed up the computation.

Let $t\in\mathcal{T}_\mathcal{I}$, and let $n := |\operatorname{row}^+(t)|$ denote the number of admissible blocks $b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$. We enumerate the corresponding column clusters as $\operatorname{row}^+(t)=\{s_1,\dots,s_n\}$.

We assume that t is not the root, i.e., there is a parent cluster $t^+ \in \mathcal{T}_{\mathcal{I}}$ such that $t \in \operatorname{chil}(t^+)$. This implies

$$\operatorname{pred}(t) = \{t\} \cup \operatorname{pred}(t^+),$$

and therefore

$$\mathcal{F}_t = \mathcal{F}_{t^+} \cup \bigcup_{j=1}^{n} \hat{s}_j.$$

Using a suitable recursive algorithm, we can assume that the matrices Z_{t^+} and Q_{t^+} and the index set $\mathcal{L}_{t^+} \subseteq \mathcal{F}_{t^+}$ with

$$G_{t^+} = V_{\mathrm{old},t^+} Z_{t^+}^* Q_{t^+}^*$$

have already been computed, and therefore we can use (6.3) to obtain

$$\begin{aligned}
G_t &= \left(G_{t^+}|_{\hat{t}\times\mathcal{F}_{t^+}} \quad V_{\mathrm{old},t} S_{\mathrm{old},t,s_1} W_{\mathrm{old},s_1}^* \quad \dots \quad V_{\mathrm{old},t} S_{\mathrm{old},t,s_n} W_{\mathrm{old},s_n}^*\right) \\
&= \left(V_{\mathrm{old},t^+}|_{\hat{t}} Z_{t^+}^* Q_{t^+}^* \quad V_{\mathrm{old},t} S_{\mathrm{old},t,s_1} W_{\mathrm{old},s_1}^* \quad \dots \quad V_{\mathrm{old},t} S_{\mathrm{old},t,s_n} W_{\mathrm{old},s_n}^*\right) \\
&= \left(V_{\mathrm{old},t} E_{\mathrm{old},t} Z_{t^+}^* Q_{t^+}^* \quad V_{\mathrm{old},t} S_{\mathrm{old},t,s_1} W_{\mathrm{old},s_1}^* \quad \dots \quad V_{\mathrm{old},t} S_{\mathrm{old},t,s_n} W_{\mathrm{old},s_n}^*\right) \\
&= V_{\mathrm{old},t} \left(E_{\mathrm{old},t} Z_{t^+}^* Q_{t^+}^* \quad S_{\mathrm{old},t,s_1} W_{\mathrm{old},s_1}^* \quad \dots \quad S_{\mathrm{old},t,s_n} W_{\mathrm{old},s_n}^*\right).
\end{aligned}$$

The first submatrix already involves an isometric factor that can be eliminated during a condensation step, but the remaining submatrices do not. QR factorizations of the column cluster basis can help us transform these matrices into a more suitable form.

Definition 7.20 (Basis weights) *A family $(R_{W,s})_{s\in\mathcal{T}_{\mathcal{J}}}$ of matrices $R_{W,s} \in \mathbb{K}^{\mathcal{L}_{W,s}\times k}$ with index sets $\mathcal{L}_{W,s} \subseteq \hat{s}$ satisfying $|\mathcal{L}_{W,s}| \leq k$ is called a family of* basis weights *for the column cluster basis W_{old} if for every $s \in \mathcal{T}_{\mathcal{J}}$ there is an isometric matrix $Q_{W,s} \in \mathbb{K}^{\hat{s}\times\mathcal{L}_{W,s}}$ such that*

$$W_{old,s} = Q_{W,s} R_{W,s}. \tag{7.14}$$

Assuming that we can construct these basis weight matrices efficiently, we can use them to write G_t in the form

$$\begin{aligned}
G_t &= V_{\mathrm{old},t} \left(E_{\mathrm{old},t} Z_{t^+}^* Q_{t^+}^* \quad S_{\mathrm{old},t,s_1} W_{\mathrm{old},s_1}^* \quad \dots \quad S_{\mathrm{old},t,s_n} W_{\mathrm{old},s_n}^*\right) \\
&= V_{\mathrm{old},t} \left(E_{\mathrm{old},t} Z_{t^+}^* Q_{t^+}^* \quad S_{\mathrm{old},t,s_1} R_{W,s_1}^* Q_{W,s_1}^* \quad \dots \quad S_{\mathrm{old},t,s_n} R_{W,s_n}^* Q_{W,s_n}^*\right) \\
&= V_{\mathrm{old},t} \left(E_{\mathrm{old},t} Z_{t^+}^* \quad S_{\mathrm{old},t,s_1} R_{W,s_1}^* \quad \dots \quad S_{\mathrm{old},t,s_n} R_{W,s_n}^*\right) \begin{pmatrix} Q_{t^+}^* & & & \\ & Q_{W,s_1}^* & & \\ & & \ddots & \\ & & & Q_{W,s_n}^* \end{pmatrix}.
\end{aligned}$$

Introducing the matrices

$$Y_t := \begin{pmatrix} Z_{t^+} E_{\mathrm{old},t}^* \\ R_{W,s_1} S_{\mathrm{old},t,s_1}^* \\ \vdots \\ R_{W,s_n} S_{\mathrm{old},t,s_n}^* \end{pmatrix} \in \mathbb{K}^{\mathcal{F}_t^c\times k}, \qquad P_t := \begin{pmatrix} Q_{t^+} & & & \\ & Q_{W,s_1} & & \\ & & \ddots & \\ & & & Q_{W,s_n} \end{pmatrix} \in \mathbb{K}^{\mathcal{F}_t\times\mathcal{F}_t^c} \tag{7.15}$$

```
**procedure** totalweights($s$, $V_{\mathrm{old}}$, $S_{\mathrm{old}}$, $R_W$, **var** $Z$, $\mathcal{L}$);
  $\mathcal{F}_t^c \leftarrow \emptyset$;
  **if** $t$ is not the root **then**
    $\mathcal{F}_t^c \leftarrow \mathcal{L}_{t^+}$;
  **for** $s \in \mathrm{row}^+(t)$ **do**
    $\mathcal{F}_t^c \leftarrow \mathcal{F}_t^c \cup \mathcal{L}_{W,s}$;
  $Y_t \leftarrow 0 \in \mathbb{K}^{\mathcal{F}_t^c \times k}$;
  **if** $t$ is not the root **then**
    $Y_t|_{\mathcal{L}_{t^+}} \leftarrow Z_{t^+} E_{\mathrm{old},t}^*$;
  **for** $s \in \mathrm{row}^+(t)$ **do**
    $Y_t|_{\mathcal{L}_{W,s}} \leftarrow R_{W,s} S_{\mathrm{old},t,s}^*$;
  Compute a QR factorization $Y_t = \widehat{Q}_t Z_t$;
  **for** $t' \in \mathrm{chil}(t)$ **do**
    totalweights($t'$, $V_{\mathrm{old}}$, $S_{\mathrm{old}}$, $R_W$, $Z$, $\mathcal{L}$)
**end**
```

Figure 7.2: Column basis weights for W_{old}

with the index set

$$\mathcal{F}_t^c := \mathcal{L}_{t^+} \cup \mathcal{L}_{W,s_1} \cup \ldots \cup \mathcal{L}_{W,s_n} \subseteq \mathcal{J},$$

we can write this identity in the short form

$$G_t = V_{\mathrm{old},t} Y_t^* P_t^*.$$

We compute a QR factorization of Y_t to obtain an isometric matrix $\widehat{Q}_t \in \mathbb{K}^{\mathcal{F}_t^c \times \mathcal{L}_t}$, a matrix $Z_t \in \mathbb{K}^{\mathcal{L}_t \times k}$ with an index set $\mathcal{L}_t \subseteq \mathcal{F}_t^c \subseteq \mathcal{J}$ satisfying $|\mathcal{L}_t| \leq k$ and

$$Y_t = \widehat{Q}_t Z_t.$$

With the isometric matrix $Q_t := P_t \widehat{Q}_t \in \mathbb{K}^{\mathcal{F}_t \times \mathcal{L}_t}$, we arrive at

$$G_t = V_{\mathrm{old},t} Y_t^* P_t^* = V_{\mathrm{old},t} Z_t^* \widehat{Q}_t^* P_t^* = V_{\mathrm{old},t} Z_t^* Q_t^*$$

and have found the required factorization.

So far, we have assumed that t is not the root. If it is, we simply skip the terms connected to t^+ in the procedure, i.e., for the root, only blocks in $\mathrm{row}^+(t)$ are considered. The resulting procedure is summarized in Figure 7.2.

It is, of course, still incomplete, since we still have to address the task of computing the basis weights $(R_{W,s})_{s \in \mathcal{T}_\mathcal{J}}$ required to set up the matrices Y_t. We can again use a recursive algorithm: if $s \in \mathcal{T}_\mathcal{J}$ is a leaf, we have $W_{\mathrm{old},s}$ at our disposal and can compute the QR factorization

$$W_{\mathrm{old},s} = Q_{W,s} R_{W,s}$$

directly, finding $R_{W,s} \in \mathbb{K}^{\mathcal{L}_{W,s} \times k}$ with an index set $\mathcal{L}_{W,s} \subseteq \hat{s}$.

```
procedure basisweights($s$, $W_{\text{old}}$, var $R_W$, $\mathcal{L}_W$);
  if chil$(t) = \emptyset$ then
    Compute a QR factorization $W_{\text{old},s} = Q_{W,s} R_{W,s}$ with $R_{W,s} \in \mathbb{K}^{\mathcal{L}_{W,s} \times k}$
  else begin
    $\widehat{\mathcal{L}}_{W,s} \leftarrow \emptyset$;
    for $s' \in$ chil$(s)$ do begin
      basisweights($s'$, $W_{\text{old}}$, $R_W$, $\mathcal{L}_W$);
      $\widehat{\mathcal{L}}_{W,s} \leftarrow \widehat{\mathcal{L}}_{W,s} \cup \mathcal{L}_{W,s'}$
    end;
    $\widehat{W}_s \leftarrow 0 \in \mathbb{K}^{\widehat{\mathcal{L}}_{W,s} \times k}$;
    for $s' \in$ chil$(s)$ do
      $\widehat{W}_s|_{\mathcal{L}_{W,s'}} \leftarrow R_{W,s'} F_{\text{old},s'}$
    Compute a QR factorization $\widehat{W}_s = \widehat{Q}_s R_{W,s}$ with $R_{W,s} \in \mathbb{K}^{\mathcal{L}_{W,s} \times k}$
  end
end
```

Figure 7.3: Column basis weights for W_{old}

Otherwise, i.e., if s has children, we can use recursion to first compute the matrices $R_{W,s'}$ for all children $s' \in \operatorname{chil}(s)$. We let $n := |\operatorname{chil}(s)|$ and $\operatorname{chil}(s) = \{s_1, \ldots, s_n\}$. Using (6.3) again, we find

$$\begin{aligned} W_{\text{old},s} &= \begin{pmatrix} W_{\text{old},s_1} F_{\text{old},s_1} \\ \vdots \\ W_{\text{old},s_n} F_{\text{old},s_n} \end{pmatrix} = \begin{pmatrix} Q_{W,s_1} R_{W,s_1} F_{\text{old},s_1} \\ \vdots \\ Q_{W,s_n} R_{W,s_n} F_{\text{old},s_n} \end{pmatrix} \\ &= \begin{pmatrix} Q_{W,s_1} & & \\ & \ddots & \\ & & Q_{W,s_n} \end{pmatrix} \begin{pmatrix} R_{W,s_1} F_{\text{old},s_1} \\ \vdots \\ R_{W,s_n} F_{\text{old},s_n} \end{pmatrix} \end{aligned}$$

We construct

$$\widehat{W}_s := \begin{pmatrix} R_{W,s_1} F_{\text{old},s_1} \\ \vdots \\ R_{W,s_n} F_{\text{old},s_n} \end{pmatrix} \in \mathbb{K}^{\widehat{\mathcal{L}}_{W,s} \times k}, \qquad \widehat{\mathcal{L}}_{W,s} := \mathcal{L}_{W,s_1} \cup \ldots \cup \mathcal{L}_{W,s_n} \subseteq \hat{s},$$

and finding a QR factorization

$$\widehat{W}_s = \widehat{Q}_{W,s} R_{W,s}$$

with an isometric matrix $\widehat{Q}_{W,s} \in \mathbb{K}^{\widehat{\mathcal{L}}_{W,s} \times \mathcal{L}_{W,s}}$ and a matrix $R_{W,s} \in \mathbb{K}^{\mathcal{L}_{W,s} \times k}$ with an index set $\mathcal{L}_{W,s} \subseteq \widehat{\mathcal{L}}_{W,s}$ with $|\mathcal{L}_{W,s}| \leq k$ yields

$$W_s = \begin{pmatrix} Q_{W,s_1} & & \\ & \ddots & \\ & & Q_{W,s_n} \end{pmatrix} \widehat{W}_s = \underbrace{\begin{pmatrix} Q_{W,s_1} & & \\ & \ddots & \\ & & Q_{W,s_n} \end{pmatrix} \widehat{Q}_{W,s}}_{=:Q_{W,s}} R_{W,s},$$

where $Q_{W,s}$ is the product of two isometric matrices and therefore isometric, too. This procedure is summarized in Figure 7.3.

Now we can get back to the task of constructing an improved cluster basis. Let $t \in \mathcal{T}_{\mathcal{I}}$. If $\operatorname{chil}(t) = \emptyset$ holds, we have $V_{\mathrm{old},t}$ at our disposal and can compute the singular value decomposition of

$$G_t^c := V_{\mathrm{old},t} Z_t^*.$$

Due to (7.13), i.e., $G_t = G_t^c Q_t^*$, the left singular vectors and singular values of this matrix are the same as of the matrix G_t, and we can proceed as in the algorithm `buildrowbasis_amatrix` to choose a rank k_t and construct $V_t \in \mathbb{K}^{\hat{t} \times k}$. Instead of computing $\widehat{G}_t$, we prepare the matrix

$$R_t := V_t^* V_{\mathrm{old},t} \in \mathbb{K}^{k_t \times k}$$

describing the change from the original to the new cluster basis.

If $\operatorname{chil}(t) \neq \emptyset$, we use recursion to prepare the cluster bases and the matrices $R_{t'}$ for all children $t' \in \operatorname{chil}(t)$. Let $n := |\operatorname{chil}(t)|$ and $\operatorname{chil}(t) = \{t_1, \ldots, t_n\}$. We have

$$\widehat{G}_t = U_t^* V_{\mathrm{old},t} Z_t^* Q_t^* = \begin{pmatrix} V_{t_1}^* V_{\mathrm{old},t_1} E_{\mathrm{old},t_1} \\ \vdots \\ V_{t_n}^* V_{\mathrm{old},t_n} E_{\mathrm{old},t_n} \end{pmatrix} Z_t^* Q_t^* = \begin{pmatrix} R_{t_1} E_{\mathrm{old},t_1} \\ \vdots \\ R_{t_n} E_{\mathrm{old},t_n} \end{pmatrix} Z_t^* Q_t^*.$$

We let

$$\widehat{V}_{\mathrm{old},t} := U_t^* V_{\mathrm{old},t} = \begin{pmatrix} R_{t_1} E_{\mathrm{old},t_1} \\ \vdots \\ R_{t_n} E_{\mathrm{old},t_n} \end{pmatrix} \tag{7.16}$$

and conclude that we have to compute the singular value decomposition of

$$\widehat{G}_t^c := \widehat{V}_{\mathrm{old},t} Z_t^*$$

and use the singular values to choose the rank k_t and the left singular vectors to define $\widehat{V}_t$ and thereby the transfer matrices of the new cluster basis. The basis change matrix can be computed efficiently via

$$R_t = V_t^* V_{\mathrm{old},t} = \widehat{V}_t^* U_t^* V_{\mathrm{old},t} = \widehat{V}_t^* \widehat{V}_{\mathrm{old},t}.$$

Since $\widehat{G}_t = \widehat{G}_t^c Q_t^*$ holds, the result is the same as in the original algorithm. The algorithm is summarized in Figure 7.4.

```
procedure buildrowbasis_h2matrix($t$, $V_{\rm old}$, $Z$, var $V$, $R$);
  if $\mathrm{chil}(t) = \emptyset$ then begin
    Compute SVD of $G_t^c := V_{\mathrm{old},t} Z_t^*$;
    Choose rank $k_t$;
    Use first $k_t$ left singular vectors to form $V_t$;
    $R_t \leftarrow V_t^* V_{\mathrm{old},t}$
  end
  else begin
    for $t' \in \mathrm{chil}(t)$ do
      buildrowbasis_h2matrix($t'$, $V_{\rm old}$, $Z$, $V$, $R$);
    Form $\widehat{V}_{\mathrm{old},t}$ according to (7.16);
    Compute SVD of $\widehat{G}_t^c := \widehat{V}_{\mathrm{old},t} Z_t^*$;
    Choose rank $k_t$;
    Use first $k_t$ left singular vectors to form $\widehat{V}_t$;
    $R_t \leftarrow \widehat{V}_t^* \widehat{V}_{\mathrm{old},t}$;
    Obtain transfer matrices by splitting $\widehat{V}_t$ according to (7.7)
  end
end
```

Figure 7.4: Adaptive row basis for an $\mathcal{H}^2$-matrix

Since the entire error analysis for the original algorithm `buildrowbasis_amatrix` remains valid for the new algorithm `buildrowbasis_h2matrix`, we only have to investigate the latter's complexity. This involves three steps: the construction of the basis weights, the construction of the total weights, and finally the construction of the new cluster basis.

Lemma 7.21 (Basis weights) *We define*

$$W_{bw}(s) := \begin{cases} C_{qr} k^2\, |\hat{s}| & \text{if } \mathrm{chil}(s) = \emptyset, \\ (C_{qr}+2) \sum_{s' \in \mathrm{chil}(s)} k^3 + W_{bw}(s') & \text{otherwise} \end{cases} \qquad \text{for all } s \in \mathcal{T}_{\mathcal{J}}.$$

Calling the function `basisweights` *for a cluster $s \in \mathcal{T}_{\mathcal{J}}$ requires not more than $W_{bw}(s)$ operations.*

Computing all basis weights takes not more than

$$C_{qr} k^2\, |\mathcal{J}| + (C_{qr}+2) k^3\, |\mathcal{T}_{\mathcal{J}}| \text{ operations.}$$

Proof. By induction over $|\mathcal{T}_s|$.

Let $s \in \mathcal{T}_{\mathcal{J}}$ with $|\mathcal{T}_s| = 1$. Then we have $\mathrm{chil}(s) = \emptyset$ and can use Assumption 5.7 to see that we need not more than

$$C_{\rm qr} |\hat{s}|\, k^2 = W_{\rm bw}(s) \text{ operations.}$$

Let now $n \in \mathbb{N}$ be such that the estimate holds for all $s \in \mathcal{T}_{\mathcal{J}}$ with $|\mathcal{T}_s| \leq n$.

Let $s \in \mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_s| = n+1$. Then we have $\operatorname{chil}(t) \neq \emptyset$ and find

$$|\widehat{\mathcal{L}}_{W,s}| \leq k\,|\operatorname{chil}(s)|.$$

Constructing $\widehat{W}_s$ requires not more than

$$\sum_{s' \in \operatorname{chil}(s)} 2|\mathcal{L}_{W,s'}|\,k^2 \leq 2k^3\,|\operatorname{chil}(s)| \text{ operations,}$$

and the QR factorization not more than

$$C_{\mathrm{qr}}|\widehat{\mathcal{L}}_{W,s}|\,k^2 \leq C_{\mathrm{qr}}k^3\,|\operatorname{chil}(s)| \text{ operations,}$$

again due to Assumption 5.7. Adding both estimates completes the induction.

For the upper bound, a simple induction yields the bound

$$\begin{aligned} C_{\mathrm{qr}} \sum_{s \in \mathcal{L}_{\mathcal{J}}} |\hat{s}|\,k^2 + (C_{\mathrm{qr}}+2) \sum_{s \in \mathcal{T}_{\mathcal{J}} \setminus \mathcal{L}_{\mathcal{J}}} \sum_{s' \in \operatorname{chil}(s)} k^3 &\leq C_{\mathrm{qr}}|\mathcal{J}|\,k^2 + (C_{\mathrm{qr}}+2) \sum_{s' \in \mathcal{T}_{\mathcal{J}}} k^3 \\ &\leq C_{\mathrm{qr}}|\mathcal{J}|\,k^2 + (C_{\mathrm{qr}}+2)|\mathcal{T}_{\mathcal{J}}|\,k^3, \end{aligned}$$

where we have used Corollary 3.19 for the first sum. ■

Lemma 7.22 (Total weights) *We define*

$$W_{tw}(t) := (C_{qr}+2)(C_{sp}+1)k^3 + \sum_{t' \in \operatorname{chil}(t)} W_{tw}(t') \qquad \textit{for all } t \in \mathcal{T}_{\mathcal{I}}.$$

Calling the function `totalweights` *for a cluster $t \in \mathcal{T}_{\mathcal{I}}$ requires not more than $W_{tw}(t)$ operations.*

Computing all total weights takes not more than

$$(C_{qr}+2)(C_{sp}+1)k^3\,|\mathcal{T}_{\mathcal{I}}| \textit{ operations.}$$

Proof. Let $t \in \mathcal{T}_{\mathcal{I}}$. We have

$$|\mathcal{F}_t^c| \leq k + k\,|\operatorname{row}^+(t)| \leq (C_{\mathrm{sp}}+1)k.$$

Constructing the matrix Y_t takes not more than

$$2k^3 + \sum_{s \in \operatorname{row}^+(t)} 2k^3 \leq 2(C_{\mathrm{sp}}+1)k^3 \text{ operations,}$$

and finding its QR factorization not more than

$$C_{\mathrm{qr}}k^2\,|\mathcal{F}_t^c| \leq C_{\mathrm{qr}}k^2(C_{\mathrm{sp}}+1)k = C_{\mathrm{qr}}(C_{\mathrm{sp}}+1)k^3 \text{ operations.}$$

Adding both estimates yields the first result. The second follows directly via a simple induction. ■

Lemma 7.23 (Improved cluster basis) *We define*

$$W_{rb}(t) := \begin{cases} (C_{svd}+4)k^2\,|\hat{t}| & \text{if } \operatorname{chil}(t)=\emptyset,\\ (C_{svd}+6)\sum_{t'\in\operatorname{chil}(t)} k^3 + W_{rb}(t') & \text{otherwise} \end{cases} \qquad \text{for all } t\in\mathcal{T}_{\mathcal{I}}.$$

Calling the function `buildrowbasis_h2matrix` *for a cluster $t\in\mathcal{T}_{\mathcal{I}}$ requires not more than $W_{rb}(t)$ operations.*

Constructing the entire adaptive row basis takes not more than

$$(C_{svd}+6)(k^2\,|\mathcal{I}| + k^3\,|\mathcal{T}_{\mathcal{I}}|) \text{ operations.}$$

Proof. By induction over $|\mathcal{T}_t|$.

Let $t\in\mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t|=1$. Then we have $\operatorname{chil}(t)=\emptyset$. Computing G_t^c requires not more than

$$2k^2\,|\hat{t}| \text{ operations,}$$

the singular value decomposition requires not more than

$$C_{\text{svd}}k^2\,|\hat{t}| \text{ operations}$$

due to Assumption 5.7, and preparing R_t takes not more than

$$2k^2\,|\hat{t}| \text{ operations.}$$

Adding the three estimates yields $W_{\text{rb}}(t)$.

Let now $n\in\mathbb{N}$ be chosen such that the estimate holds for all $t\in\mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t|\le n$.

Let $t\in\mathcal{T}_{\mathcal{I}}$ with $|\mathcal{T}_t|=n+1$. Then we have $\operatorname{chil}(t)\neq\emptyset$. Constructing $\widehat{V}_{\text{old},t}$ requires not more than

$$\sum_{t'\in\operatorname{chil}(t)} 2k^3 \text{ operations.}$$

Since this matrix hat not more than

$$\sum_{t'\in\operatorname{chil}(t)} k = k\,|\operatorname{chil}(t)|$$

rows, we can compute $\widehat{G}_t^c$ in not more than

$$2k^3\,|\operatorname{chil}(t)| \text{ operations}$$

and Assumption 5.7 yields that not more than

$$C_{\text{svd}}k^3\,|\operatorname{chil}(t)| \text{ operations}$$

are required for the singular value decomposition. The matrix R_t can finally be computed in not more than

$$2k_tk^2\,|\operatorname{chil}(t)| \le 2k^3\,|\operatorname{chil}(t)| \text{ operations,}$$

and adding the four estimates yields $W_{\text{rb}}(t)$, completing the induction.

For the upper bound, a simple induction yields

$$\begin{aligned}(C_{\text{svd}}+4)\sum_{t\in\mathcal{L}_{\mathcal{I}}} k^2\,|\hat{t}| + (C_{\text{svd}}+6)\sum_{t\in\mathcal{T}_{\mathcal{I}}\setminus\mathcal{L}_{\mathcal{I}}}\sum_{t'\in\operatorname{chil}(t)} k^3 \\ \leq (C_{\text{svd}}+4)k^2\,|\mathcal{I}| + (C_{\text{svd}}+6)\sum_{t'\in\mathcal{T}_{\mathcal{I}}} k^3 \\ \leq (C_{\text{svd}}+4)k^2\,|\mathcal{I}| + (C_{\text{svd}}+6)k^3\,|\mathcal{T}_{\mathcal{I}}|,\end{aligned}$$

where we have used Corollary 3.19 for the first sum. ■

We can see that $\mathcal{O}(k^2|\mathcal{J}| + k^3|\mathcal{T}_{\mathcal{J}}|)$ operations are sufficient to compute the basis weights and that $\mathcal{O}(k^2|\mathcal{I}| + k^3|\mathcal{T}_{\mathcal{I}}|)$ operations are sufficient for the total weights and the improved row basis.

Remark 7.24 (Typical complexity) *If we have $|\hat{t}| \geq k$ and $|\hat{s}| \geq k$ for all leaves $t \in \mathcal{L}_{\mathcal{I}}$ and $s \in \mathcal{L}_{\mathcal{J}}$ and if no cluster has exactly one child, we have already seen in Remark 6.10 that we have $k|\mathcal{T}_{\mathcal{I}}| \leq 2|\mathcal{I}|$ and $k|\mathcal{T}_{\mathcal{J}}| \leq 2|\mathcal{J}|$ and find that our algorithms require $\mathcal{O}(k^2\,|\mathcal{J}|)$ and $\mathcal{O}(k^2\,|\mathcal{I}|)$ operations, respectively, i.e., we obtain linear complexity.*

Remark 7.25 (Weighted estimates) *Since the new algorithm works with the total weights Z_t instead of the original matrices G_t, we cannot choose arbitrarily general weighting strategies as in Remark 7.13.*

Fortunately, there is still enough flexibility to ensure blockwise error estimates: if we let $\omega_{t,s} := \|G|_{\hat{t}^+\times\hat{s}}\|_2 q^{\operatorname{level}(t)-\operatorname{level}(t^+)}$ for $t \in \mathcal{T}_{\mathcal{I}}$, $t^+ \in \operatorname{pred}(t)$, and $s \in \operatorname{row}^+(t^+)$, we can see that we only have to modify the definition of Y_t in (7.15) by including simple weights:

$$Y_{t,\omega} := \begin{pmatrix} Z_{t^+}E^*_{old,t}/q \\ R_{W,s_1}S^*_{old,t,s_1}/\|G|_{\hat{t}\times\hat{s}_1}\|_2 \\ \vdots \\ R_{W,s_n}S^*_{old,t,s_n}/\|G|_{\hat{t}\times\hat{s}_n}\|_2 \end{pmatrix}.$$

Of course, now we have to compute the norms of the admissible submatrix blocks. If we prepare basis weights R_V for the row cluster basis as well as R_W for the column cluster basis, we have

$$\|G|_{\hat{t}\times\hat{s}}\|_2 = \|V_{old,t}S_{old,t,s}W^*_{old,s}\|_2 = \|R_{V,t}S_{old,t,s}R^*_{W,s}\|_2 \qquad \text{for all } b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},$$

and the matrix on the right has not more than k rows and columns, therefore its spectral norm can be computed efficiently. Since we need only a lower bound, we can even use a power iteration to estimate this norm.

Bibliography

[1] M. Bebendorf. Approximation of boundary element matrices. *Numer. Math.*, 86(4):565–589, 2000.

[2] M. Bebendorf. *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems.* LNCSE. Springer, 2008.

[3] M. Bebendorf and R. Grzhibovskis. Accelerating Galerkin BEM for linear elasticity using adaptive cross approximation. *Math. Meth. Appl. Sci.*, 29:1721–1747, 2006.

[4] M. Bebendorf and S. Rjasanow. Adaptive low-rank approximation of collocation matrices. *Computing*, 70(1):1–24, 2003.

[5] S. Börm. Adaptive variable-rank approximation of dense matrices. *SIAM J. Sci. Comp.*, 30(1):148–168, 2007.

[6] S. Börm. *Efficient Numerical Methods for Non-local Operators: $\mathcal{H}^2$-Matrix Compression, Algorithms and Analysis*, volume 14 of *EMS Tracts in Mathematics.* EMS, 2010.

[7] S. Börm and L. Grasedyck. Hybrid cross approximation of integral operators. *Numer. Math.*, 101:221–249, 2005.

[8] S. Börm and W. Hackbusch. Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices. *Computing*, 69:1–35, 2002.

[9] A. Cohen, W. Dahmen, and R. DeVore. Adaptive wavelet methods for elliptic operator equations — Convergence rates. *Math. Comp.*, 70:27–75, 2001.

[10] J. W. Cooley and J. W. Tukey. An algorithm for the machine computation of complex Fourier series. *Math. Comp.*, 19:297–301, 1965.

[11] W. Dahmen, H. Harbrecht, and R. Schneider. Compression techniques for boundary integral equations — Asymptotically optimal complexity estimates. *SIAM J. Numer. Anal.*, 43(6):2251–2271, 2006.

[12] W. Dahmen, S. Prössdorf, and R. Schneider. Multiscale methods for pseudo-differential equations on smooth manifolds. In C. K. Chui, L. Montefusco, and L. Puccio, editors, *Proceedings of the International Conference on Wavelets: Theory, Algorithms and Applications*, pages 385–424. Academic Press, San Diego, 1994.

[13] W. Dahmen, S. Prössdorf, and R. Schneider. Wavelet approximation methods for pseudodifferential equations I: Stability and convergence. *Math. Z.*, 215:583–620, 1994.

[14] W. Dahmen and R. Schneider. Wavelets on manifolds I: Construction and domain decomposition. *SIAM J. Math. Anal.*, 31:184–230, 1999.

[15] R. A. DeVore and G. G. Lorentz. *Constructive Approximation.* Springer-Verlag, 1993.

[16] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1:211–218, 1936.

[17] G. H. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis*, 2(2):205–224, 1965.

[18] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, London, 1996.

[19] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin. A theory of pseudoskeleton approximations. *Lin. Alg. Appl.*, 261:1–22, 1997.

[20] L. Grasedyck. *Theorie und Anwendungen Hierarchischer Matrizen.* Doctoral thesis, Universität Kiel, 2001.

[21] L. Grasedyck and W. Hackbusch. Construction and arithmetics of $\mathcal{H}$-matrices. *Computing*, 70:295–334, 2003.

[22] L. Greengard and V. Rokhlin. A new version of the fast multipole method for the Laplace equation in three dimensions. In *Acta Numerica 1997*, pages 229–269. Cambridge University Press, 1997.

[23] W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing*, 62(2):89–108, 1999.

[24] W. Hackbusch. *Hierarchical Matrices: Algorithms and Analysis.* Springer, 2015.

[25] W. Hackbusch and B. N. Khoromskij. A sparse $\mathcal{H}$-matrix arithmetic: General complexity estimates. *J. Comp. Appl. Math.*, 125:479–501, 2000.

[26] W. Hackbusch and B. N. Khoromskij. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part II: Application to multi-dimensional problems. *Computing*, 64:21–47, 2000.

[27] W. Hackbusch, B. N. Khoromskij, and S. A. Sauter. On $\mathcal{H}^2$-matrices. In H. Bungartz, R. Hoppe, and C. Zenger, editors, *Lectures on Applied Mathematics*, pages 9–29. Springer-Verlag, Berlin, 2000.

[28] L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *Quart. J. Math. Oxford. Ser.*, 11(2):50–59, 1960.

[29] T. J. Rivlin. *The Chebyshev Polynomials.* Wiley-Interscience, New York, 1990.

[30] S. A. Sauter. Variable order panel clustering. *Computing*, 64:223–261, 2000.

[31] J. Stoer. *Einführung in die Numerische Mathematik I.* Springer, 5th edition edition, 1989.

[32] E. E. Tyrtyshnikov. Mosaic-skeleton approximation. *Calcolo*, 33:47–57, 1996.

[33] E. E. Tyrtyshnikov. Incomplete cross approximation in the mosaic-skeleton method. *Computing*, 64:367–380, 2000.

Index