Chapter 13
$\mathcal{H}$-LU Iteration

Abstract The $\mathcal{H}$-LU iteration is a fast iteration for discretisations of boundary value problems. It even applies to fully populated matrices obtained by the boundary element method. The $\mathcal{H}$-LU iteration has an almost optimal order of convergence. Section 13.1 describes computing the general LU decomposition by using hierarchical matrices. In the case of sparse matrices, in particular, finite element matrices, the cluster tree can be modified (cf. Section 13.2) so that the corresponding LU decomposition partially preserves sparsity. The $\mathcal{H}$-LU decomposition is not exact, but the error can be rather small. Correspondingly, the $\mathcal{H}$-LU iteration described in Section 13.4 is very fast. The variant discussed in §13.4.2 is purely algebraic, i.e., the data needed for the iteration are only based on the underlying matrix. Concerning details about the technique of hierarchical matrices, we refer to Appendix D.

13.1 Approximate LU Decomposition

The LU decomposition based on the technique of hierarchical matrices (cf. Appendix D) yields an approximation of the exact LU factors in $A = LU$ and is called $\mathcal{H}$-LU decomposition (cf. Hackbusch [198, §7.6 and §7.8] and Grasedyck–Kriemann–Le Borne [165, 166]). The accuracy can be controlled by an appropriate local rank. Therefore the $\mathcal{H}$-LU factorisation is quite different from the incomplete (ILU) decomposition described in §7.3.

In the positive definite case, LU decomposition can be replaced by Cholesky decomposition $A = LL^{\mathsf{T}}$. For symmetric but not necessarily positive definite matrices A, we may use the LDL decomposition $A = LDL^{\mathsf{T}}$. In the following we restrict ourselves to the general LU case. In this section we make no assumption about sparsity of the matrix. The algorithms explained below can also be applied to fully populated matrices, e.g., arising from discretising an integral equation.

In Section 13.2 we shall assume that A is a sparse finite element matrix. Then it is largely possible to preserve sparsity, i.e., the computed factors L and U contain many vanishing matrix blocks.

W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*,
Applied Mathematical Sciences 95, DOI 10.1007/978-3-319-28483-5_13

13.1.1 Triangular Matrices

Triangular matrices can only be defined with respect to a prescribed ordering. The appropriate ordering of the index set I is described in §D.2.1.3. The ordering is consistent with $T(I)$ since each cluster $\tau \in T(I)$ contains consecutive indices:

$$\tau = \{i_{\alpha(\tau)}, i_{\alpha(\tau)+1}, \ldots, i_{\beta(\tau)}\}. \tag{13.1}$$

Correspondingly, disjoint clusters τ, σ are ordered: $\tau < \sigma$ holds if $i < j$ for all $i \in \tau$ and $j \in \sigma$. Let $M \in \mathcal{H}(r, P)$ be a hierarchical matrix (cf. Definition D.12). All blocks $b = \tau \times \sigma \in P$ with $\tau \neq \sigma$ are lying completely in the strictly upper (U) or lower triangular part (L). Diagonal blocks $\tau \times \tau \in P$ belong to P^- and the corresponding matrix blocks $M|_{\tau\times\tau}$ are represented as full matrices.

The definition of the format of the hierarchical triangular matrices L and U of the LU decomposition is that they be triangular and hierarchical:

$$L, U \in \mathcal{H}(r, P), \quad \begin{cases} L_{i_\alpha i_\beta} = 0 & \text{for } \alpha < \beta, \\ L_{i_\alpha i_\alpha} = 1 & \text{for } 1 \leq \alpha \leq \#I, \\ U_{i_\alpha i_\beta} = 0 & \text{for } \alpha > \beta. \end{cases} \tag{13.2}$$

Solvability of a system $LUx = b$ requires that $U_{i_\alpha i_\alpha} \neq 0$ for all i_α.

The triangular matrices can also be replaced by *block-triangular matrices*:

$$\begin{array}{lllll} \text{off-diagonal blocks:} & L|_{\tau\times\sigma} = O & \text{for } \tau < \sigma & \text{and} & U|_{\tau\times\sigma} = O \ \text{ for } \tau > \sigma, \\ \text{diagonal blocks:} & L|_{\tau\times\tau} = I & \text{and} & U|_{\tau\times\tau} \in \mathcal{F}(\tau\times\tau) & \text{for } \tau\times\tau \in P. \end{array}$$

Concerning the restriction $\cdot|_b$ to a block b see (A.8b) or Notation D.6. Note that $U|_{\tau\times\tau}$ is no longer triangular. The block-triangle decomposition has the advantage that it may be well defined even if the standard LU decomposition does not exist.

13.1.2 Solution of $LUx = b$

Given a factorisation $A = LU$, the system $Ax = b$ is solved in two stages: the equation $Ly = b$ is treated by the procedure *Forward_Substitution* and $Ux = y$ by *Backward_Substitution*. These steps can easily be formulated for hierarchical matrices and performed exactly. The procedure *Forward_Substitution*(L, τ, y, b) yields the (exact) solution $y|_\tau$ of $L|_{\tau\times\tau} y|_\tau = b|_\tau$. To solve $Ly = b$, one has to call *Forward_Substitution*(L, I, y, b) with $\tau = I$ (the input vector b is overwritten).

```
**procedure** *Forward_Substitution*$(L, \tau, y, b)$;
**if** $\tau \times \tau \in P$ **then for** $j := \alpha(\tau)$ **to** $\beta(\tau)$ **do**          (cf. (13.1))
        **begin** $y_j := b_j$; **for** $i := j+1$ **to** $\beta(\tau)$ **do** $b_i := b_i - L_{ij} y_j$ **end**
**else for** $j := 1$ **to** $\#S(\tau)$ **do**
**begin** *Forward_Substitution*$(L, \tau[j], y, b)$;
        **for** $i := j+1$ **to** $\#S(\tau)$ **do** $b|_{\tau[i]} := b|_{\tau[i]} - L|_{\tau[i]\times\tau[j]} \cdot y|_{\tau[j]}$
**end**;
```

The requirements for the parameters are: $\tau \in T(I \times I, P)$, $y, b \in \mathbb{K}^I$, and L satisfies (13.2) with $P \subset T(I \times I)$. In line 6, $\tau[1], \ldots, \tau[\#S(\tau)]$ is an enumeration of the sons of τ.

The procedure *Backward_Substitution* for solving $Ux = y$ is quite similar. U, τ, y are input parameters, while x is the output. The vector y is overwritten.

```
**procedure** *Backward_Substitution*$(U, \tau, x, y)$;
**if** $\tau \times \tau \in P$ **then for** $j := \beta(\tau)$ **downto** $\alpha(\tau)$ **do**
    **begin** $x_j := y_j / U_{jj}$;
        **for** $i := \alpha(\tau)$ **to** $j - 1$ **do** $y_i := y_i - U_{ij} x_j$
    **end**
**else for** $j := \#S(\tau)$ **downto** $1$ **do**
**begin** *Backward_Substitution*$(U, \tau[j], x, y)$;
    **for** $i := 1$ **to** $j - 1$ **do** $y|_{\tau[i]} := y|_{\tau[i]} - U|_{\tau[i] \times \tau[j]} \cdot x|_{\tau[j]}$
**end**;
```

The complete solution of $LUx = b$ uses

```
**procedure** *Solve_LU*$(L, U, I, x, b)$;        $\{L, U, I, b$ input; $x$ output$\}$
**begin** $x := b$;
    *Forward_Substitution*$(L, I, x, x)$;
    *Backward_Substitution*$(U, I, x, x)$
**end;**
```

Formulating the block version is left to the reader as an exercise. Since then the diagonal matrix blocks $U|_{\tau \times \tau}$ ($\tau \times \tau \in P$) must be inverted during the solution of $Ux = y$, the best approach is to invert $U|_{\tau \times \tau}$ immediately after constructing U. Then the backward substitution procedure can multiply by the precomputed inverse stored in $U|_{\tau \times \tau}$.

Finally, we need an algorithm for solving $x^{\mathsf{T}} U = y^{\mathsf{T}}$. This equation is identical to $Lx = y$ with $L := U^{\mathsf{T}}$; however, in this case, the lower triangular matrix L is not normed. The corresponding procedure is left to the reader:

procedure *Forward_SubstitutionT*(U, τ, x, y); $\{$solving $x^{\mathsf{T}} U = y^{\mathsf{T}}\}$.

13.1.3 Matrix-Valued Solutions of $LX = Z$ and $XU = Z$

The matrix $L \in \mathcal{H}(r, P)$ with $P \subset T(I \times I)$ is a lower triangular matrix (cf. (13.2)). Let $X, Z \in \mathcal{H}(r, P')$ be rectangular hierarchical matrices corresponding to a partition $P' \subset T(I \times J)$. The index set I is the same as for $L \in \mathbb{K}^{I \times I}$. We want to solve the matrix equation

$$LX = Z$$

in $\mathbb{K}^{I\times J}$, which represents $\#J$ simultaneous equations of the form $Lx = z$. The following procedure *Forward_M* solves $L|_{\tau\times\tau}X|_{\tau\times\sigma} = Z|_{\tau\times\sigma}$ for the blocks $\tau\times\tau \in T(I\times I, P)$ and $\tau\times\sigma \in T(I\times J, P')$. The complete system $LX = Z$ in $I \times J$ is solved by *Forward_M*(L, X, Z, I, J). By $X_{\tau,j}$ we denote the j-th columns of $X \in \mathbb{K}^{\tau\times\sigma}$, i.e., $X_{\tau,j} = (X_{ij})_{i\in\tau}$.

```
procedure Forward_M$(L, X, Z, \tau, \sigma)$;
if $\tau\times\sigma \in P^-$ then                {column-wise forward substitution}
    for all $j \in \sigma$ do Forward_Substitution$(L, \tau, X_{\tau,j}, Z_{\tau,j})$
else if $\tau\times\sigma \in P^+$ then
begin   {let $Z|_{\tau\times\sigma} = AB^\mathsf{T}$ according to (D.1) with $A \in \mathbb{K}^{\tau\times\{1,\ldots,r\}}$}
    for $j = 1$ to $r$ do Forward_Substitution$(L, \tau, A'_{\tau,j}, A_{\tau,j})$;
    $X|_{\tau\times\sigma}$ := rank-$r$ representation by $A'B^\mathsf{T}$
end else
for $i = 1$ to $\#S(\tau)$ do for $\sigma' \in S(\sigma)$ do
begin Forward_M$(L, X, Z, \tau[i], \sigma')$;
    for $j = i + 1$ to $\#S(\tau)$ do    {$\ominus, \odot$: operations with truncation}
    $Z|_{\tau[j]\times\sigma'} := Z|_{\tau[j]\times\sigma'} \ominus L|_{\tau[j]\times\tau[i]} \odot X|_{\tau[i]\times\sigma'}$
end;
```

In the standard case of $\#S(\sigma) = 2$, the problem

$$L|_{\tau\times\tau}X|_{\tau\times\sigma} = Z|_{\tau\times\sigma}$$

has the block structure

$$\begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix}$$

with

$$L_{ij} = L|_{\tau[i]\times\tau[j]}, \quad X_{ij} = X|_{\tau[i]\times\sigma[j]}, \quad Z_{ij} = Z|_{\tau[i]\times\sigma[j]}.$$

The equations $L_{11}X_{11} = Z_{11}$ and $L_{11}X_{12} = Z_{12}$ of the first block row are solved for $i = 1$ by the call of *Forward_M* in line 10, whereas the remaining equations $L_{21}X_{11} + L_{22}X_{21} = Z_{21}$ of the first block row and $L_{21}X_{12} + L_{22}X_{22} = Z_{22}$ of the second one are reformulated as

$$L_{22}X_{21} = Z'_{21} := Z_{21} - L_{21}X_{11}, \quad L_{22}X_{22} = Z'_{22} := Z_{22} - L_{21}X_{12}$$

in line 12 and are solved for $i = 2$ in line 10 with respect to X_{21}, X_{22}.

For solving the equation $XU = Z$ with an upper triangular hierarchical matrix U and an unknown matrix X left of U, we use a similar procedure involving the procedure *Forward_SubstitutionT* defined above:

$$\textbf{procedure}\ \textit{ForwardT\_M}\ (U, X, Z, \tau, \sigma); \tag{13.3}$$

(details in [198, (7.33b)]).

13.1.4 Generation of the LU Decomposition

It remains to describe the generation of the hierarchical LU factors in

$$A = LU \in \mathbb{K}^{I\times I}.$$

To simplify the explanation we assume that $\#S(I) = 2$. Then the matrices in $A = LU$ have the structure

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & O \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ O & U_{22} \end{bmatrix}. \tag{13.4}$$

This leads to the four subtasks

(i) compute L_{11} and U_{11} as factors of the LU decomposition of A_{11},
(ii) compute U_{12} from $L_{11}U_{12} = A_{12}$,
(iii) compute L_{21} from $L_{21}U_{11} = A_{21}$,
(iv) compute L_{22} and U_{22} as LU decomposition of $L_{22}U_{22} = A_{22} - L_{21}U_{12}$.

Problem (ii) is solved by the procedure *Forward_M*$(L_{11}, U_{12}, A_{12}, \tau_1, \tau_2)$, whereas for problem (iii) we use the procedure *ForwardT_M* in (13.3). The right-hand side in

$$L_{22}U_{22} = A_{22} - L_{21}U_{12}$$

can be computed by the usual formatted multiplication.

We still have to determine the LU factors of $L_{11}U_{11} = \ldots$ and $L_{22}U_{22} = \ldots$. This defines a recursion, which at the leaves is defined by the usual LU decomposition of full matrices.

The call of *LU_Decomposition*(L, U, A, I) yields the desired LU factors of A. More generally, the procedure *LU_Decomposition*(L, U, A, τ) solves the problem $L|_{\tau\times\tau}U|_{\tau\times\tau} = A|_{\tau\times\tau}$ for $\tau \in T(I \times I, P)$.

```
procedure LU_Decomposition$(L,U,A,\tau)$ ;
if $\tau\times\tau\in P$ then compute $L|_{\tau\times\tau}$ and $U|_{\tau\times\tau}$ as LU factors of $A|_{\tau\times\tau}$
else for $i=1$ to $\#S(\tau)$ do
begin LU_Decomposition$(L,U,A,\tau[i])$ ;
    for $j=i+1$ to $\#S(\tau)$ do
    begin ForwardT_M $(U,L,A,\tau[j],\tau[i])$;
          Forward_M$(L,U,A,\tau[i],\tau[j])$;
          for $r=i+1$ to $\#S(\tau)$ do
             $A|_{\tau[j]\times\tau[r]} := A|_{\tau[j]\times\tau[r]} \ominus L|_{\tau[j]\times\tau[i]} \odot U|_{\tau[i]\times\tau[r]}$
end end;
```
(13.5)

The sons of $S(\tau)$ are denoted by $\tau[1], \ldots, \tau[\#S(\tau)]$.

13.1.5 Cost of the $\mathcal{H}$-LU Decomposition

Because of the triangular structure, the two matrices L and U need not more storage than a usual hierarchical matrix:

$$S_{\mathrm{LU}}(r,P) = S_{\mathcal{H}}(r,P),$$

where $S_{\mathcal{H}}(r,P)$ is given in Lemma D.17.

As in Lemma D.18, one verifies that the cost of *Forward_Substitution*(L,I,y,b) can be estimated by the double storage cost of L. An analogous result holds for *Backward_Substitution*(U,τ,x,y). Together we obtain

$$N_{\mathrm{LU}}(r,P) \le 2S_{\mathcal{H}}(r,P).$$

Comparing the costs for solving both systems $LX = Z$ and $XU = Z$ with a standard multiplication of hierarchical matrices, we obtain

$$N_{\mathrm{Forward\_M}}(r,P) + N_{\mathrm{ForwardT\_M}}(r,P) \le N_{\mathrm{MM}}(P,r,r)$$

with $N_{\mathrm{MM}}(P,r,r)$ in (D.15). Generating the LU decomposition by the procedure in (13.5) also does not require more operations than matrix-matrix multiplication:

$$N_{\text{LU decomposition}}(r,P) \le N_{\mathrm{MM}}(P,r,r).$$

13.2 $\mathcal{H}$-LU Decomposition for Sparse Matrices

13.2.1 Finite Element Matrices

Finite element matrices are *sparse* in the classical sense. They can exactly be transferred into the $\mathcal{H}(r,P)$ format. This transfer is required if we want to apply hierarchical matrix operations other than matrix-vector multiplication.

Lemma 13.1. *Let $\mathcal{H}(r,P) \subset \mathbb{K}^{I\times I}$ be an arbitrary hierarchical format, and P an admissible partition. Let $\operatorname{dist}(\tau,\sigma)$ be defined by (D.8) and (D.9b). Then any finite element matrix belongs to $\mathcal{H}(r,P)$ for all $r \in \mathbb{N}_0$.*

Proof. For an admissible block $b = \tau\times\sigma \in P^+$, the indices $i \in \tau$ and $j \in \sigma$ belong to basis functions with disjoint supports X_i and X_j. Hence the finite element matrix restricted to b is a zero block and, therefore, belongs to $\mathcal{R}_r(b)$. □

Modern direct solvers for sparse systems apply sophisticated algorithms to minimise the fill-in during the LU decomposition. Formally, this means finding a permutation P, so that the LU decomposition of PAP^{T} (without pivoting) is sparser than for A. For instance, one may try to minimise the band width since fill-in of the LU factors occurs only within the band (cf. [314, §3.9.1]). Similarly, we may try to optimise the $\mathcal{H}$-LU decomposition for sparse matrices.

The precise conditions concerning the sparsity pattern will be discussed in §13.2.2. Let I be the index set in $A \in \mathbb{K}^{I\times I}$. The ordering of the index set, determining the LU decomposition, is derived from the cluster tree $T(I)$ (cf. (13.1)). Therefore, alternative permutations require alternative cluster trees. Such a cluster tree will be introduced in §13.2.3. The following LU variants are based on the articles Le Borne–Grasedyck–Kriemann [259] and Grasedyck–Kriemann–Le Borne [166].

The inverse of a sparse finite element matrix is a fully populated matrix. It is shown in Bebendorf–Hackbusch [39], Faustmann–Melenk–Praetorius [129], and Faustmann [128] that the inverse matrix can be well approximated by the format $\mathcal{H}(r,P)$. The involved truncation error decreases exponentially with r. These results can be transferred to the LU decomposition; i.e., the factors L and U are also well approximated by hierarchical triangular matrices in $\mathcal{H}(r,P)$ (cf. [198, §9.2.8] or Grasedyck–Kriemann–Le Borne [166], Faustmann [128, §6]). A similar result holds for the inverse and the LU factors of matrices arising from the boundary element method (cf. Faustmann–Melenk–Praetorius [130]).

13.2.2 Separability of the Matrix

Sparsity alone is not sufficient for our purpose. In addition, we need the following condition. The index set I can be decomposed disjointly:

$$I = I_1 \,\dot\cup\, I_2 \,\dot\cup\, I_s \quad \text{with} \quad \#I_1 \approx \#I_2,\ \#I_s \ll \#I, \tag{13.6a}$$

so that the matrix A, which we want to partition, has the following block structure:

$$A = \begin{array}{c|c|c|c} & I_1 & I_2 & I_s \\ \hline I_1 & A_{11} & O & A_{1s} \\ \hline I_2 & O & A_{22} & A_{2s} \\ \hline I_s & A_{s1} & A_{s2} & A_{ss} \end{array} . \tag{13.6b}$$

The index set I_s is called the *separator* since $A|_{(I\backslash I_s)\times(I\backslash I_s)}$ is decomposed into the matrix blocks A_{11} and A_{22}; the off-diagonal blocks A_{12} and A_{21} contain only zero entries.

Condition $\#I_1 \approx \#I_2$ in (13.6a) ensures that (i) A_{11} and A_{22} be similar in size, (ii) the zero blocks are large.

Condition $\#I_s \ll \#I$ requires the separator to be comparably small. More quantitative statements will follow.

The requirements (13.6a,b) can easily be formulated by the matrix graph $G(A)$ (cf. §C.2). I is the vertex set. There must be a (small) subset I_s so that the graph without the I_s-vertices and corresponding edges disaggregates into two unconnected subgraphs with the vertex sets I_1 and I_2 (cf. Fig. 13.1).

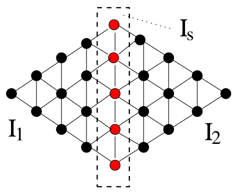

Fig. 13.1 Matrix graph separated by I_s.

The last formulation yields a sufficient condition for (13.6a,b). If $G(A)$ is a planar graph, a linear subgraph—as in Figure 13.1—is a sufficient choice of the separator. Planar graphs are, e.g., obtained by discretising two-dimensional boundary value problems by a standard difference method or by piecewise linear finite elements. If $n = \#I$ is the problem size, one expects a separator of the cardinality $\#I_s = \mathcal{O}(\sqrt{n})$, while $\#I_1, \#I_2 \approx n/2$. In the case of finite elements in a domain $\Omega \subset \mathbb{R}^2$, one determines a curve $\gamma \subset \overline{\Omega}$ with endpoints on $\Gamma = \partial\Omega$, consisting of edges belonging to the finite element triangulation (cf. Fig. 13.2). The indices $i \in I_s$ are associated with the nodal points in γ. The vertices left or right of γ form the respective sets I_1 or I_2. If $i_1 \in I_1$ and $i_2 \in I_2$, supports of the basis functions ϕ_{i_1} and ϕ_{i_2} lie on different sides of γ and can overlap at most by their boundaries. This implies that $A_{i_1 i_2} = 0$, as required in (13.6b).

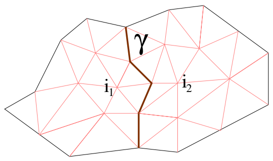

Fig. 13.2 Domain decomposition by γ.

The example of a boundary value problem in Ω shows that the method can be iterated: γ divides Ω into subdomains Ω_1 and Ω_2, and the submatrices A_{11} and A_{22} in (13.6b) belong to boundary value problems in these subdomains; hence, they are of the same kind as the original matrix.

The latter observation leads to the final assumption:

$$\text{The submatrices } A_{ii} := A|_{I_i \times I_i}\ (i = 1,2) \text{ must again satisfy (13.6a–c) or be sufficiently small.} \tag{13.6c}$$

This requirement ensures that the partition can be continued recursively (Fig. 13.3 shows the result after two partitions). Obviously, the condition $\#I_s \ll \#I$ is vague. In particular, the symbol $\ll$ is meaningless if $\#I$ is not large. In this case, the recursion terminates since 'sufficiently small' submatrices occur (cf. (13.6c)).

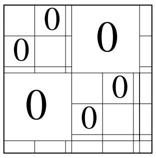

Fig. 13.3 Twofold decomposition.

The partition (13.6a,b) is well known as the *dissection* method introduced by George [149]. It also corresponds to the (iterated form of the) domain decomposition method.

13.2.3 Construction of the Cluster Tree

The partition of the index set I into the three subsets in (13.6a) can easily be performed. A variant of the partition in §D.2.1.2 works as follows. Assume that the indices $i \in I$ are again associated with nodal points $\xi_i \in \mathbb{K}^d$. Let the partition of the cuboid (minimal box) yield the binary decomposition of I into $\hat{I}_1$ and $\hat{I}_2$. The first set $I_1 := \hat{I}_1$ remains unchanged, while the second is split again:

$$I_s := \{i \in \hat{I}_2 : \text{there are } A_{ij} \neq 0 \text{ or } A_{ji} \neq 0 \text{ for some } j \in I_1\}, \qquad I_2 := \hat{I}_2 \backslash I_s.$$

Obviously, the partition into I_1, I_2, I_s satisfies condition (13.6a).

In principle, this decomposition algorithm could be continued recursively. The result would be a ternary tree $T(I)$. However, this procedure is not optimal. The reason are the different characters of the three subsets I_1, I_2, and I_s. For an illustration, assume the two-dimensional case $\Omega \subset \mathbb{R}^2$. The first two sets I_1 and I_2 correspond to the (two-dimensional) subdomains Ω_1 and Ω_2 (cf. Fig. 13.2), whereas the indices of I_s are vertices of the (one-dimensional) curve γ. We recall the bisection of the bounding box in §D.2.1.2. d bisection steps of a d-dimensional cuboid lead to 2^d subcuboids of half the size. This means that the diameter of an index set belonging to subdomains of Ω is reduced by about $1/\sqrt{2}$, whereas the diameter of an index set belonging to the (one-dimensional) separator γ is reduced by $1/2$. Therefore, with increasing level ℓ, the subset $T^{(\ell)}(I)$ defined in (D.7) contains index sets exhibiting increasingly different sizes. Therefore the block cluster tree contains rather flat blocks $\sigma \times \tau$.

The following modification (here explained and illustrated for $d = 2$) avoids a systematic distortion of the cluster sizes in $T^{(\ell)}(I)$. The cluster set $T(I)$ is divided into 'two-dimensional' clusters $T_d(I)$ and 'one-dimensional' clusters $T_{d-1}(I)$. Their definition is given by

(a) $I \in T_d(I)$,

(b) if $\tau \in T_d(I)$, the sons τ_1, τ_2 belong to $T_d(I)$, whereas τ_s belongs to $T_{d-1}(I)$,

(c) all successors of $\tau \in T_{d-1}(I)$ belong to $T_{d-1}(I)$.

In Figure 13.4, the rectangles with dashed sides correspond to clusters in $T_{d-1}(I)$, the other rectangles correspond to $T_{d-1}(I)$.

The decomposition rules are as follows:

(a) A cluster $\tau \in T_d(I)$ is always decomposed into three parts. Since, in the case of an LU decomposition, an ordering of the sons of τ is required, we define the order as follows: First, the sons $\tau_1, \tau_2 \in S(\tau) \cap T_d(I)$ are arranged in arbitrary order (edges depicted by solid lines in Fig. 13.4), then the son $\tau_s \in S(\tau) \cap T_{d-1}(I)$ follows (dashed line).

(b) The treatment of a cluster $\tau \in T_{d-1}(I)$ depends on its graph distance to the next $T_d(I)$-predecessor. For this purpose, we introduce

$$\varkappa(\tau) := \min\{\text{level}(\tau) - \text{level}(\tau') : \tau' \in T_d(I) \text{ predecessor of } \tau\}.$$

(ba) If $\varkappa(\tau)$ is odd, τ remains unchanged (dotted edge in Fig. 13.4).

(bb) If $\varkappa(\tau)$ is even, τ is decomposed in a binary[1] way according to §D.2.1.2 (broken-dotted edges in Fig. 13.4).

These rules guarantee that all clusters in $T^{(\ell)}(I)$ have successors at level $\ell + 2$ with a diameter of about half the size. For $d = 3$, one has to modify these rules suitably.

[1] Here a ternary splitting does not make sense.

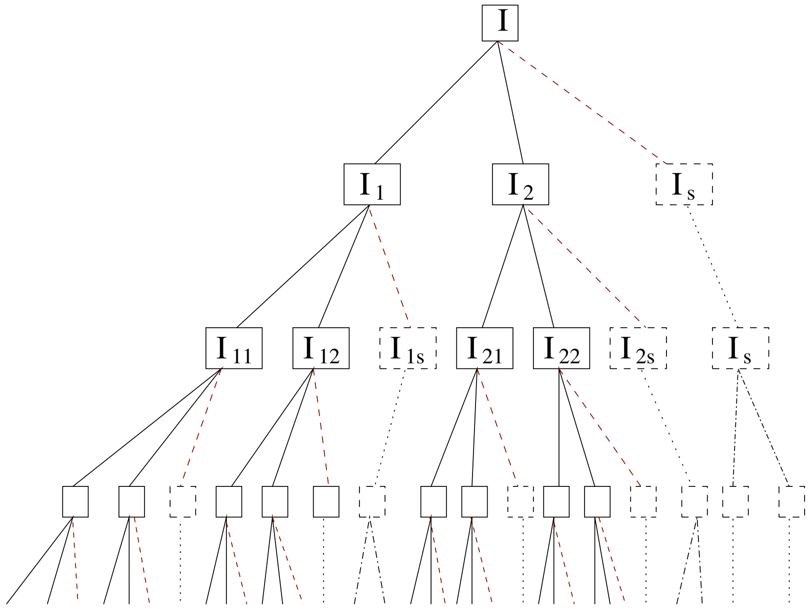

Fig. 13.4 Cluster tree $T(I)$.

The corresponding block cluster tree $T(I \times I)$ is obtained as in Definition D.8. A block partition of depth $L = 2$ is shown in Figure 13.3.

13.2.4 Application to Inversion

The inversion algorithm in §D.3.6 has an intrinsic disadvantage concerning its parallel treatment. The inversion of $M|_{\tau\times\tau}$ has to wait until the inversions in the blocks $\tau' \times \tau'$ $(\tau' \in S(\tau))$ are performed. This requires a sequential computing[2]. Also in the case of partition (13.6b), one has first to invert the diagonal blocks A_{11} and A_{22} before the Schur complement in $I_s \times I_s$ can be formed and inverted, but (i) the inverses of A_{11} and A_{22} can be computed in parallel and (ii) the computations in $I_s \times I_s$ are significantly cheaper than the inversions of A_{11} and A_{22} because of $\#I_s \ll \#I$.

The algorithm is still sequential in the level-number: The inversion of $M|_{\tau\times\tau}$ can take place as soon as the inversions in $\tau' \times \tau'$ $(\tau' \in S(\tau))$ are performed.

More details about this method can be found in Hackbusch [192] and Hackbusch–Khoromskij–Kriemann [202]. Parallel $\mathcal{H}$-matrix implementations are discussed by Kriemann [245].

[2] Of course, the arising matrix-matrix multiplications and additions can be parallelised.

13.2.5 Admissibility Condition

The zero blocks in (13.6b) are characterised by

$$\tau' \times \tau'' \text{ with } \tau' \neq \tau'' \text{ and } \tau', \tau'' \in S(\tau) \cap T_d(I) \quad \text{for some } \tau \in T_d(I). \qquad (13.7)$$

The blocks $b = \tau' \times \tau''$ are not admissible in the sense of Definition D.11, since the support sets $X_{\tau'}$ and $X_{\tau''}$ touch at the separating line γ, and therefore $\operatorname{dist}(\tau', \tau'')$ vanishes. Nevertheless, it does not make sense to decompose b again. Therefore the admissibility condition adm^* in (D.11) is modified as follows:

$$\operatorname{adm}^{**}(\tau' \times \tau'') := \left[\operatorname{adm}^*(\tau' \times \tau'') \text{ or } \tau' \times \tau'' \text{ satisfies (13.7)}\right].$$

The minimal admissible partition $P \subset T(I \times I)$ is now defined in (D.12) with adm^* replaced by adm^{**}. So far, we divided P into the near- and far-field: $P = P^- \dot\cup P^+$. Now a ternary partition is appropriate: $P = P^0 \dot\cup P^- \dot\cup P^+$ with $P^0 := \{b \in P \text{ satisfies (13.7)}\}$, while $P \backslash P^0$ is split into $P^- \dot\cup P^+$ as before.

13.2.6 LU Decomposition

The algorithm in §13.1 can be applied without changes. The advantage of the new cluster tree $T(I)$ can be seen from the following statement.

Remark 13.2. Let the matrix $A \in \mathcal{H}(r, P)$ satisfy $A|_b = 0$ for all $b \in P^0$. Then the approximate LU decomposition according to (13.5) yields factors L, $U \in \mathcal{H}(r, P)$ satisfying again $L|_b = U|_b = 0$ for $b \in P^0$ (cf. Fig. 13.5).

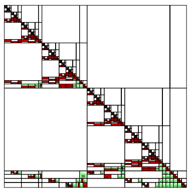
Fig. 13.5 Factor U; white blocks are zero.

Detailed numerical results and comparisons with other algorithms can be found in Grasedyck–Hackbusch–Kriemann [164].

13.3 UL Decomposition of the Inverse Matrix

If a regular matrix A possesses an LU decomposition $A = LU$, A^{-1} can also be decomposed into $U'L'$ with $L' := L^{-1}$ and $U' := U^{-1}$ and vice versa. Here we use that the inverse of a (normed) triangular matrix is again (normed) triangular. Note the different ordering of the factors in $A^{-1} = U'L'$: the first matrix is the upper triangular, while the second one is the normed lower matrix.

Remark 13.3. The standard forward and backward substitution in $x \mapsto U^{-1}L^{-1}x$ avoids inversion, but is mainly sequential. In contrast to this, matrix-vector multiplications in $x \mapsto U'L'x$ can be parallelised much better (see Kriemann–Le Borne [246, Tables 3 and 4]).

Similar to the discussion of (13.4), the factors in $A^{-1} = U'L'$ can be determined from $L'AU' = I$ (cf. [198, §7.6.5] and Kriemann–Le Borne [246]).

13.4 $\mathcal{H}$-LU Iteration

13.4.1 General Construction

The $\mathcal{H}$-matrix technique may be considered as a direct method with the difference that the error is not characterised by the machine precision, but by the accuracy of the $\mathcal{H}$-matrix computation. Note that the $\mathcal{H}$-matrix accuracy can be adjusted to the discretisation error.

In fact, there is a smooth transition from a direct method to an iterative one. We recall that even the Gauss elimination becomes an iteration when it is re-iterated (cf. Skeel [341], Björck [48, §1.4.6]).

The $\mathcal{H}$-LU decomposition $A \approx LU$ induces the iteration $\Phi_{\mathcal{H}\text{-LU}}$:

$$x^{m+1} = x^m - W^{-1}(Ax^m - b) \qquad \text{with } W = LU. \tag{13.8}$$

The properties of the method are collected in the next remark.

Remark 13.4. (a) Since an LU decomposition does not exist for any regular matrix, the existence of the $\mathcal{H}$-LU decomposition is not guaranteed in general. If the hierarchical LU decomposition is successful, the involved rank controls the error $I - W^{-1}A$.

(b) The inversion of $W = LU$ uses the procedures in §13.1.2, which are very fast.

(c) The data required to determine W are the matrix A including the geometric information about the nodal points ξ_i $(i \in I)$. In the case of a sparse matrix, the geometric data can be replaced by the graph $G(A)$. In that case, the method is algebraic (cf. Definition 2.2b).

(d) If $A > 0$, also $W > 0$ is expected (here Cholesky decomposition should be used). There are strategies to ensure $W > 0$ in spite of truncation errors (cf. [198, §6.8.2]). Then the iteration is positive definite: $\Phi_{\mathcal{H}\text{-LU}} \in \mathcal{L}_{\text{pos}}$.

The statements $W \approx A$ or $N = W^{-1} \approx A^{-1}$ can be made more precise by the error estimate

$$\|I - NA\|_2 \le \varepsilon < 1. \tag{13.9a}$$

Inequality (13.9a) implies the corresponding estimate with respect to the spectral radius:

$$\rho(I - NA) \le \varepsilon < 1. \tag{13.9b}$$

If, e.g., $\varepsilon = \frac{1}{10}$ in (13.9a), each step of the iteration (13.8) improves the result by one decimal. $\varepsilon = \frac{1}{10}$ is already considered as fast convergence, whereas N with $\varepsilon = \frac{1}{10}$ in (13.9a) may be still regarded as a rough approximation of the inverse.

An alternative to (13.9a) is

$$\|I - NA\|_A = \|I - A^{1/2}NA^{1/2}\|_2 \le \varepsilon < 1 \tag{13.9c}$$

for positive definite A. (13.9c) also implies (13.9b). The contraction properties (13.9a) or (13.9c) are very important if only a few iteration steps are performed.

For determining the approximate inverse $N = W^{-1}$, we have to weight up the following properties.

- *Relatively rough approximation (moderate $\varepsilon < 1$):* In this case, a smaller local rank of the $\mathcal{H}$-matrix representation is sufficient; hence, the storage cost and computational cost is reduced. As a consequence, we have to perform several steps of the iterative method (13.8). However, the latter fact is of lesser importance since the matrix-vector multiplications Ax^m and Nd for $d := Ax^m - b$ are significantly faster than inversion or LU decomposition required for N.
- *Relatively accurate approximation (small $\varepsilon \ll 1$):* The local rank of the $\mathcal{H}$-matrix representation will increase logarithmically with $1/\varepsilon$. On the other hand, the iterative method requires only one or two steps.

The effective amount of work is

$$\mathrm{Eff}(\Phi_{\mathcal{H}\text{-LU}}) = \mathcal{O}(r^\alpha \log^\beta n) \qquad (n = \#I)$$

with $\alpha, \beta > 0$ (cf. (2.31a)). Therefore smaller local ranks r may be preferred. Usually, the maximal r is $\mathcal{O}(\log^* n)$, since then the discretisation error is reached. Hence the effective amount of work is always bounded by $\mathcal{O}(\log^* n)$; i.e., the $\mathcal{H}$-LU iteration is almost optimal.

Let A and W be a positive definite matrix. We recall the spectral equivalence of A and W defined by

$$\frac{1}{c}\langle Ax, x\rangle \leq \langle Wx, x\rangle \leq c\,\langle Ax, x\rangle \qquad \text{for all } x \in \mathbb{K}^I \tag{13.10}$$

with a constant $c > 0$ (cf. Definition 7.56). According to (7.51e), (13.10) is equivalent to $\frac{1}{c}I \leq A^{-1/2}WA^{-1/2} \leq cI$. Inversion yields $\frac{1}{c}I \leq A^{1/2}NA^{1/2} \leq cI$. Applying (13.9c) yields the next statement.

Remark 13.5. Inequality (13.9c) implies the spectral equivalence (13.10) with

$$c := \frac{1}{1-\varepsilon} \approx 1 + \varepsilon.$$

The spectral equivalence may come into play by other means. The solution of nonlinear problems or parabolic differential equations can lead to the situation[3] that many systems $A^{(\nu)}x^{(\nu)} = b^{(\nu)}$ are to be solved, involving different matrices $A^{(\nu)}$ which are still spectrally equivalent. Then it is sufficient to approximate the inverse $N = (A^{(0)})^{-1}$ of the first matrix and to use this approximation as preconditioner for all $A^{(1)}, A^{(2)}, \ldots$ (cf. page 321).

[3] For instance, in the nonlinear case the Newton method leads to different linearisations $A^{(\nu)}$, where ν is the index of the Newton iteration. In the parabolic case, the matrices $A(t)$ depend on the time t. The time steps $t = 0, \Delta t, 2\Delta t, \ldots$ yield $A^{(\nu)} = A(\nu\Delta t)$.

13.4.2 Algebraic LU Decomposition

The LU decomposition described in §13.1 is still dependent on geometric data (coordinates of the nodal points). The following construction removes this dependence and uses only data contained in the matrix A, provided that A is a sparse matrix.

Given A, we obtain the graph $G(A)$ (cf. §C.2). More precisely, we use the undirected graph $G := G_{\mathrm{sym}}(A) = G(A) \cup G(A^{\mathsf{T}})$. We assume that G is connected, since otherwise the system decomposes into at least two separated systems. For a connected graph, any two $\alpha, \beta \in I$ are connected by at least one path. The path length is defined by the number of edges between α and β. The minimal length of all paths between α and β defines the distance $\delta(\alpha, \beta)$.

The distance δ yields the necessary topology. It allows defining the diameter of a cluster and the distance of two clusters. Therefore, admissibility condition (D.10) can be formulated.

The construction of the cluster tree $T(I)$ and, in particular, of the special cluster tree corresponding to §13.2.3 is explained in [198, §9.2] and Grasedyck–Kriemann–Le Borne [165]. The latter paper contains numerical examples which show that this approach is rather robust.

13.5 Further Applications of Hierarchical Matrices

The 'commonly used matrix formulation' $Ax = b$ in (1.5) is not the only representation. The linear equation may take the form of a *matrix equation*. For this purpose, let $A, C \in \mathbb{K}^{n\times n}$ be given matrices, while $X \in \mathbb{K}^{n\times n}$ is an unknown matrix. Then

$$AX + XA^{\mathsf{H}} = C \tag{13.11}$$

represents the Lyapunov equation. This is a system of linear equations for all entries of X. In principle, we can form vectors $x, c \in \mathbb{K}^{n^2}$ and a matrix $\mathcal{A} \in \mathbb{K}^{n^2\times n^2}$ such that (13.11) is equivalent to $\mathcal{A}x = c$. However, if A is a large-scale matrix, the size of x is equal to n^2 which may be too large for practical computations. The remedy is to use a format for the matrix X which involves only $\mathcal{O}(n)$ or $\mathcal{O}(n \log^* n)$ data. Hierarchical matrices are a possible choice. Possibly, even global low-rank matrices can be used. As shown by Penzl [310], $\mathrm{rank}(C) = r$ implies that the singular values of X decrease exponentially. This property ensures approximability by global low-rank matrices.

A slight generalisation is the Sylvester equation $AX + XB = C$ with given matrices A, B, and C. Corresponding statements and approximations by global low-rank and hierarchical matrices are discussed by Grasedyck [159], Baur [35], Baur–Benner [36]), and Benner–Breiten [40].

Even the (nonlinear) Riccati equation $AX + XA^{\mathsf{H}} - XBX = C$ can be solved (cf. Hackbusch [198, §15.2] and Grasedyck–Hackbusch–Khoromskij [163]).