Order Number 9432384

# Parallel algorithms for banded linear systems of equations

Sobze, Isdor, Ph.D.

City University of New York, 1994

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

# Parallel Algorithms for Banded Linear Systems of Equations

by

Isdor Sobze

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

1994

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

4/20/94 — Date

Victor Pan — Chair of Examining Committee

April 20, 1994 — Date

Executive Officer

Wayne Eberly

Robert Orchard

Bogong Su

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

PARALLEL ALGORITHMS FOR BANDED LINEAR SYSTEMS OF EQUATIONS

by

Isdor Sobze

Adviser: Professor Victor Y. Pan

**Abstract.** We devise *parallel algorithms* for solving a nonsingular *banded linear system* of equations (BLS) and for computing the determinant of any *banded matrix*, substantially improving the previous record computational complexity estimates of W. Eberly [E]. These algorithms are in $NC$ (respectively, $RNC$) if the input matrix is defined over a field of characteristic zero (respectively, a finite field or a ring); they support new record bounds on the *parallel time complexity* of these computations and the *optimum* or *near optimum* bounds on their *potential work* (that is, the product of time and processor bounds); moreover, they are in $NC^1$ or $RNC^1$ if the *bandwidth* of the input matrix is a constant. We also develop several preprocessing techniques for accelerating the computation of the solution of several nonsingular BLS's, each having the same coefficient matrix $A$, so as to keep the potential work on solving $q$ such BLS's asymptotically bounded by the potential work on solving a single BLS, provided that $q$ does not exceed the bandwidth of the common input matrix $A$. The space requirement of our solutions is equal (up to a small multiplicative constant factor) to the space required by the best known sequential algorithms for solving the same problems.

## ACKNOWLEDGMENTS

The main idea leading to the work presented in this dissertation was suggested by Professor Victor Y. Pan (my thesis advisor), who also provided, besides the necessary advice, the main financial support through his PSC CUNY and NSF grants, during both the research and the writing phases of this thesis.

I am grateful to the CUNY College of Staten Island (CSI) for the teaching fellowship, an important financial support which enabled me to complete the required course work and to study for the qualifying examination. I particularly thank Professors Charles Giardina, Robert Orchard and Miriam Tausner, who fought, as much as they could, to keep my financial assistance line open and who have always given me their unconditional love and moral support. I feel more like a family member of each one of them.

A number of classmates have directly or indirectly contributed to the research leading to the results presented in this dissertation. In particular, Antoine Atinkpahoun, Steve Yu, Akim Sadikou and Elliot Landowne. I thank each one of them for their participation and useful comments.

After reviewing this thesis, my examining committee (Professors Wane Eberly, Robert Orchard, Stanley Habib, Bogong Su) has made some suggestions and comments that have led to some modifications in the presentation of this work. I want to thank each of the above listed members for servicing

in the committee and for taking the time to reading this material.

Last, but not least, I would like to thank my wife Shelley and my daughter Dora, for accepting several years of loneliness. Though occasionally fussy (when the paper mess got tough), they were with me up to the end.

# Contents

# List of Figures

# Chapter 1

# Introduction and background.

## 1.1 Introduction.

> The ancient scientist/philosopher such as Aristotle, who was able to comprehend almost all knowledge available in his time, has gradually been replaced by generations of scientists with ever increasing depth of knowledge and narrowness of interest and competence... Limitation of human mind seems to be the primary reason of this trend ... Once the amount of knowledge is greater than the human mind is able to comprehend, any increase in the knowledge necessary means that a human comprehends a small fraction of it [Kl, page 3].

Due to the advances in parallel computer architecture, machines with large numbers of processors[1] are now available. Due to this technological progress, some researchers predict that "within a decade, all developments in computer applications and algorithm design will be taking place within the context of parallel computation" [KR].

In this thesis, we have primarily directed our efforts to the design of *parallel algorithms* for solving a nonsingular *banded linear system of equations*

[1] Parallel machines with more than 64K processors have been constructed already about a decade ago (see [Qui]).

and for computing the determinant of a *banded matrix.* A matrix is said to be *banded* if all its nonzero entries are located in a small band centered around its principal diagonal. A linear system is called banded if its coefficient matrix is banded. Such linear systems frequently arise in practice of scientific and engineering computations and has long been a subject of intensive research (see [Am], [LP], [DB, page 165], [Ba, page 343], [GL, pages 149–159, 170–177]); also see further bibliography in [GL, pages 158–159]. We seek parallel solutions that are *efficient* or (preferably) *optimal* (the concepts of efficiency and optimality, in the context of parallel computations, are discussed in §1.4). This dissertation includes the recent algorithms and techniques from [PSA]. Some of these banded matrix algorithms (in particular, the ones of chapter 5) may be extended to any matrix having the (block) Hessenberg form, provided that this matrix has lower (block) *edge* (see Definition 1.3.2, page 10).

We organize our presentation in the following order: In this chapter, we recall some relevant concepts of *linear algebra* and *parallel computations*, formalize our main goals and summarize the main results. In §1.2, we review some basic definitions and some fundamental operations of matrix algebra. In §1.3, we recall the customary terminology for *banded linear systems* of equations and discuss some block matrix representations of a banded matrix, in particular, the ones relevant to the design of our algorithms. Our high level description of the algorithms is machine independent; to measure their parallel complexity estimates we will assume general models of parallel

computing under the *parallel random access machine (PRAM)* which we recall in §1.4, together with some recent results on parallel (dense) matrix computations. In §1.5, we state the main problems of our study. In §1.6, we apply the classical Gaussian elimination to solving a banded linear system. This technique does not lead to *efficient* parallel algorithms for solving this problem[2]; however, it is one of the simplest and most utilized sequential algorithms which supports the record bounds on the sequential computational complexity of the solution of a nonsingular banded linear system and the evaluation of the determinant of a banded matrix. We will compare the bound on its sequential complexity with the bound on the *potential work* (that is, on the product of time and processor bounds) of our parallel algorithms for the same computational problems. In §1.7, we summarize the approaches of [PSA] to the solution and state our main results.

In chapter 2 we present optimum parallel algorithms for the above problems, over the fields of characteristic zero, based on *block cyclic reduction.*

In chapters 3 and 4, we present alternative solutions to the same problems; these solutions are efficient (but, not optimal); however, they are applicable over any field of constants. The algorithms of chapter 3 rely on a $3 \times 3$ representation and a recursive factorization of the input matrix. A similar approach was already utilized by W. Eberly [E] to establish the record

[2]Gaussian elimination combined with an appropriate reordering of the variables and equations of the input linear system may lead to an *optimal parallel algorithm* for solving banded linear systems over the fields of characteristic zero, but then, the technique is called *nested dissection* (see [Ge73], [GeLi], [LRT79], [PR93]).

bounds on the parallel computational complexity of solving a banded linear system of equations. We show an improvement of this record bound by a factor $O(\log^2 n)$. In chapter 4, we present an independent solution approach, based on a $2 \times 2$ factorization of the input matrix, which supports the same asymptotic computational complexity bounds (as in chapter 3) but requires fewer random parameters.

In chapter 5, we assume that our input matrix has lower and/or upper *edge* (see Definition 1.3.2, page 10). Under this (mild) restriction, we derive solutions which are applicable over any field of constants, deterministic, and require fewer fields operations than all other known fast parallel algorithms for the same problems.

In chapter 6, we summarize our main results and discuss some related open problems.

## 1.2 Some basic definitions.

> The block matrix concept is very important from the standpoint of both theory and practice. On the theoretical side, the block matrix notation allows us to prove important matrix factorizations, very succinctly. These factorizations are the cornerstones of numerical linear algebra. From the computational point of view, block matrix algorithms are important, because they are rich in matrix multiplication, the operation of choice for many new high performance computer architectures [GL, page 1].

All the algorithms presented in this dissertation utilize block matrices (that is, matrices with matrix entries). As has been pointed out in the above comments from [GL], this concept gives us a powerful tool for the design

and correctness proof of matrix algorithms. Moreover, the block matrix notation enables us to operate at a high level. Furthermore, by operating with matrix blocks (rather than scalars), the matrix algorithm designer has better control on the overhead due to the data movement, which leads in practice to algorithms having better performance (see [GL, page 25]).

**Notation.** Hereafter, $\mathbf{F}$ denotes an arbitrary field, $\mathbf{C}$ the field of complex numbers, the upper case letters $A$, $B$, ..., $Z$ the matrices over any field of constants, $(a_{ij})_{1\le i\le m,\ 1\le j\le n}$ an $m \times n$ matrix $A$, $0_{m,n}$ the $m \times n$ null matrix, $0_n$ the $n \times n$ null matrix, $0$ the null matrices (when the size is understood from the context), $I_k$ the $k \times k$ identity matrix, $\mathrm{diag}\,(a_1, a_2, \ldots, a_n)$ the $n \times n$ diagonal matrix where $a_i$ is the $(i,i)$ entry, $W^T$ the transpose of a matrix $W$, $W^H$ the Hermitian transpose[3] of a matrix $W$ over the field $\mathbf{C}$, $\vec{\mathbf{x}}$ a column vector $(x_1, \ldots, x_n)^T$, $(A_{ij})_{1\le i\le p,\ 1\le j\le q}$ a $p \times q$ block matrix with $m_i \times n_j$ blocks $A_{ij}$, of the format

$$A = \begin{pmatrix} A_{11} & A_{12} & \ldots & A_{1q} \\ A_{21} & A_{22} & \ldots & A_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ A_{p1} & A_{p2} & \ldots & A_{pq} \end{pmatrix} \begin{matrix} m_1 \\ m_2 \\ \vdots \\ m_p. \end{matrix} \qquad (1.1)$$

$$\begin{matrix} n_1 & n_2 & \ldots & n_q \end{matrix}$$

Many operations with scalar matrices are immediately extended to block matrices.

We let $\mathrm{diag}\,(A_1, A_2, \ldots, A_p)$ denote the $p \times p$ block diagonal matrix, where

[3]Some authors prefer the notation $W^*$ for the matrix $W^H$.

each $(i,i)$ entry $A_i$ is a square matrix; the addition, subtraction and multiplication of block matrices are defined (similarly to the ones on scalar matrices) as long as the obvious block size requirements are met (see [GL, pages 27–25] for further details).

REMARK **1.2.1** For numerical matrix computations, it is customary and sufficient to work over the field **C** of complex numbers or its real or rational subfields.

A linear system of $m$ equations and $n$ unknowns over a field **F**

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad 1 \le i \le m, \tag{1.2}$$

will be represented by using the customary notation

$$A\vec{\mathbf{x}} = \vec{\mathbf{b}}, \tag{1.3}$$

where $A = (a_{ij})$, $\vec{\mathbf{x}} = (x_1,\ldots,x_n)^T$ and $\vec{\mathbf{b}} = (b_1,\ldots,b_n)^T$, or, alternatively, by means of $p$ vector equations of the format

$$\sum_{j=1}^{q} A_{ij}\vec{\mathbf{x}}_j = \vec{\mathbf{b}}_i, \quad 1 \le i \le p, \tag{1.4}$$

where $(A_{ij})$ is a $p \times q$ block matrix representation of $A$ with $m_i \times n_j$ blocks, $\sum_{i=1}^{p} m_i = m$, $\sum_{j=1}^{q} n_j = n$, $\vec{\mathbf{x}}_j$ is an $n_j$-dimensional (column) vector, $\vec{\mathbf{b}} = \left(\vec{\mathbf{b}}_1^T,\ldots,\vec{\mathbf{b}}_p^T\right)^T$, each $\vec{\mathbf{b}}_i$ has dimension $m_i$.

DEFINITION **1.2.1 (Nonsingularity and inverse matrix).** An $n \times n$ matrix $A$ is *nonsingular* if $AX{=}XA{=}I_n$, for some $n \times n$ matrix $X$; the matrix

$X$ (if it exists) is unique; it is called the *inverse* of $A$ and denoted $A^{-1}$. If a matrix is not nonsingular, it is said to be *singular.*

FACT **1.2.1** If $A$ and $B$ are nonsingular, then the product $AB$ is nonsingular and $(AB)^{-1} = B^{-1}A^{-1}$.

DEFINITION **1.2.2 (Permutation matrix).** An $n \times n$ matrix $P$ is called a *permutation matrix* if, for any $n$-dimensional column vector $\overrightarrow{\mathbf{v}} = (v_1, \ldots, v_n)^T$, there exists a permutation

$$\pi : \{1, \ldots, n\} \longrightarrow \{1, \ldots, n\}$$

such that $P\overrightarrow{\mathbf{v}} = (v_{\pi(1)}, \ldots, v_{\pi(n)})^T$ and $\overrightarrow{\mathbf{v}}^T P^T = (v_{\pi(1)}, \ldots, v_{\pi(n)})$.

DEFINITION **1.2.3 (Submatrix).** A $k \times l$ matrix formed by the intersection of the rows $i_1, \ldots, i_k$ and the columns $j_1, \ldots, j_l$ of a matrix $W$, for any $k$-tuple $(i_1, \ldots, i_k)$ and any $l$-tuple $(j_1, \ldots, j_l)$ , is a *submatrix* of $W$.

A $k \times k$ submatrix of a matrix $W$, formed by rows and columns $i_1, \ldots, i_k$ of $W$, for any $k$-tuple $(i_1, \ldots, i_k)$, is called *principal.*

DEFINITION **1.2.4 (Strong nonsingularity).** A matrix is *strongly nonsingular* if all its principal submatrices are nonsingular.

DEFINITION **1.2.5** A matrix $A$ is *Hermitian* if $A^H = A$. A matrix is *Hermitian nonnegative definite* (*h.n.d.* ) if it can be represented as $W^H W$, for some matrix $W$. A nonsingular h.n.d. matrix is called *Hermitian positive definite* (*h.p.d.* ).

PROPOSITION **1.2.1** (see [GL, page 140]). *In the field of complex numbers (and in any of its subfields), the matrices $W^HW$ and $(W^HW)^{-1}$ are strongly nonsingular for any nonsingular matrix $W$.*

REMARK **1.2.2 (Symmetrization technique).** Over the field $\mathbf{C}$ of complex numbers or any subfield of $\mathbf{C}$, the problem of solving any nonsingular linear system of equations, having coefficient matrix $W$, can be reduced to solving an h.p.d. linear system, having coefficient matrix $W^HW$. Consequently, it is theoretically sufficient to develop linear system solvers for the special case of h.p.d. linear systems (see §2.3, §3.5 and §4.5).

DEFINITION **1.2.6 (Determinant).** The *determinant* of an $n \times n$ matrix $A$, denoted $\det A$, is defined by the relation

$$\det A = \begin{cases} a_{11}, & \text{if } A = (a_{11}); \\ \sum_{j=1}^{n} (-1)^{j+1} a_{ij} \det A_{ij}, & \text{otherwise,} \end{cases} \tag{1.5}$$

where $A_{ij}$ denotes the $(n-1) \times (n-1)$ submatrix of $A$ obtained by deleting the $i$th row and the $j$th column from $A$.

PROPOSITION **1.2.2** ( see [GL, page 52] or [Ba, page 60]). *Let $\alpha \in \mathbf{F}$, $A$ and $B$ be two $n \times n$ matrices over $\mathbf{F}$, $P$ and $Q$ be two $p \times p$ block matrices over $\mathbf{F}$ with $b \times b$ blocks.*

1. $\det(AB) = (\det A)(\det B)$; $\det A^T = \det A$; $\det(\alpha A) = \alpha^n \det A$.
2. *$A$ is nonsingular* $\Leftrightarrow \det A \neq 0 \Leftrightarrow \det(A^{-1}) = 1/\det A$.

3. *If $B$ is a matrix resulting from swapping two rows (or columns) of $A$, then* $\det B = -\det A$.

4. *If $B$ is a matrix resulting from adding to any row (respectively, column) of $A$ a linear combination of other rows (respectively, columns) of $A$, then* $\det B = \det A$.

5. *If $Q$ is a matrix resulting from swapping two block rows (or columns) of $P$, then* $\det Q = (-1)^b \det P$.

6. *If $Q$ is a matrix resulting from adding to any block row (respectively, block column) of $P$, a linear combination of other block rows (respectively, block columns) of $P$, then* $\det Q = \det P$.

## 1.3 Banded linear systems (BLS).

DEFINITION **1.3.1 (Bandwidth of a matrix).** A matrix $A = (a_{ij})$ has *lower* (respectively, *upper*) *bandwidth* $w_- = w_-(A)$ (respectively, $w_+ = w_+(A)$) if $w_-$ (respectively, $w_+$) is the minimum nonnegative integer such that $a_{ij} = 0$ for $i > j + w_-$ (respectively, $j > i + w_+$). The sum $w = w_+ + w_- = w(A)$ is called the *bandwidth* of $A$ (see figure 1.1).

For examples, if $A$ is a diagonal matrix, then $w(A) = 0$; if $A$ is a full $n \times n$ triangular matrix, then $w(A) \leq n - 1$; if $A$ is a full $n \times n$ matrix, then $w(A) \leq 2(n - 1)$;

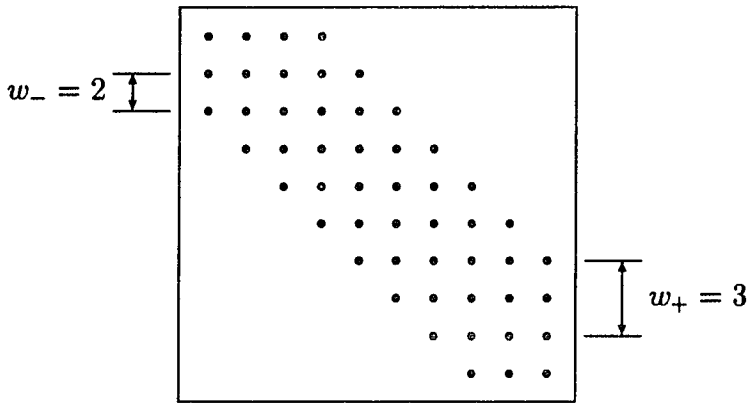



Figure 1.1: A $10 \times 10$ matrix with bandwidth $w = 5$.

---

A $k \times h$ matrix $A$ is called *banded* when $k$ and $h$ largely exceed its bandwidth $w(A)$.

DEFINITION **1.3.2 (Upper and lower edges).** In chapter 5, we will utilize the following concept: a matrix $A$ has *lower* (respectively, *upper*) *edge*, if $a_{ij} \neq 0$ whenever $i = j + w_-$ (respectively, $j = i + w_+$).

PROPOSITION **1.3.1** *$w(W^H W) \leq 2w(W)$ and $w(W_k) \leq w(W)$ for any principal submatrix $W_k$ of $W$.*

FACT **1.3.1 (Block tridiagonal representation of a banded matrix).** Let $A$ be an $n \times n$ matrix and let $w$ be any integer such that

$$\max\{w_-(A), w_+(A)\} \leq w < n;$$

then the matrix $A$ has a $p \times p$ block matrix representation with $w \times w$ blocks[4] of the following format where $p = \lceil n/w \rceil$:

$$A = \begin{pmatrix} A_{00} & A_{01} & & & \\ A_{10} & A_{11} & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & A_{p-2,p-2} & A_{p-2,p-1} \\ & & & A_{p-1,p-2} & A_{p-1,p-1} \end{pmatrix}. \tag{1.6}$$

Note that, conversely, block tridiagonal matrices are also banded.

Due to the block tridiagonal representation of a banded matrix, the algorithms for solving block tridiagonal linear systems immediately apply to banded linear systems. Several of these algorithms are based on *block cyclic reduction* (see [GL], [BD], [BGN], [He], [Sw] [Swe74] or [Swe77]), which can be extended to an optimal parallel algorithm for solving any nonsingular banded linear systems over the fields of characteristic zero (see chapters 2 and 5).

FACT **1.3.2 (Representation-2).** Any $n \times n$ matrix having bandwidth $w$, where $n > 2w$, has a block representation of the format

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \tag{1.7}$$

where the blocks $A_{11}$ and $A_{22}$ have sizes $n_1 \times n_1$ and $n_2 \times n_2$, respectively, and where $n_1 = \lceil n/2 \rceil$, $n_2 = n - n_1$, $A_{12} = \begin{pmatrix} 0 & 0 \\ L & 0 \end{pmatrix}$, $A_{21} = \begin{pmatrix} 0 & U \\ 0 & 0 \end{pmatrix}$, $U$ and $L$ are $w \times w$ matrices. We will refer to (1.7) as to *representation-2* of a matrix $A$.

[4] If $w$ does not divide $n$, the last block row (respectively, column) may have a size smaller than that of other block rows (respectively, columns).

FACT **1.3.3 (Representation-3).** Any $n \times n$ matrix having bandwidth $w$ where $n \geq 3w$ has a block representation of the format

$$A = \begin{pmatrix} A_{11} & A_{12} & 0 \\ A_{21} & A_{22} & A_{23} \\ 0 & A_{32} & A_{33} \end{pmatrix}, \tag{1.8}$$

where the blocks $A_{11}, A_{22}$ and $A_{33}$ have sizes $n_1 \times n_1$, $w \times w$ and $n_3 \times n_3$, respectively, where $n_1 = \lceil (n-w)/2 \rceil$, $n_3 = n - w - n_1$, $A_{12} = \begin{pmatrix} 0 \\ S \end{pmatrix}$, $A_{21} = (0 \quad E)$, $A_{23} = (W \quad 0)$, $A_{32} = \begin{pmatrix} N \\ 0 \end{pmatrix}$, $N$, $S$, $E$, $W$ are $w \times w$ matrices. We will refer to (1.8) as to *representation-3* of a matrix $A$.

## 1.4 The *PRAM* models of parallel computing.

> "The *PRAM* cannot be considered a physically realizable model, since, as the number of processors and the size of the global memory scales up, it quickly becomes impossible to provide a constant-length data path from any processor to any memory cell. Nevertheless, the *PRAM* has proven to be an extremely useful vehicle for studying the logical structure of parallel computation in a context divorced from issues of parallel communication... Algorithms developed for other, more realistic models are often based on algorithms originally designed for the *PRAM*" [KR].

### 1.4.1 Terminology.

There are several abstract models that may be utilized as a theoretical basis for parallel computations, such as the *circuit* model, the *fixed connection networks* and the *parallel random access machine* (*PRAM*) models.

A *circuit* (see [Re93, page 11]) is a directed acyclic graph (DAG), with ordered edges and labeled nodes derived from a *straight line program*, by

applying the following rules: 1) the variables are renamed so that every non-input variable is assigned only once; 2) the nodes of the circuit are exactly the distinct variables of the resulting straight line program; 3) the input nodes correspond to the input variables; 4) each non-input node $v$ is labeled with the operation used in its corresponding variable assignment, and has an ordered sequence of edges entering $v$ and departing from the sequence of nodes on whose values this assignment depends. The *time* for parallel evaluation of the circuit is its *depth*. The *size* of the circuit is the number of its non-input nodes.

A *fixed connection network* (see [Re93, page 12]) is a collection of processors working in parallel. The processors are the nodes of a fixed digraph; these processors communicate with each other via the edges of the digraph. This model has led to several practical parallel architectures, including the *hypercube* (that is, a $d$-dimensional array of $n$ nodes, where $d = \log n$) and the *butterfly* network.

A *parallel random access machine* (also known as *shared memory machine*) consists of several independent processors, each having its own *local memory*, communicating with each other through a *global (shared) memory*. In one unit of time, each processor can read one (global or local) memory location, execute a single *RAM* operation ([AHU, page 5]) and write into one (global or local) memory location (see figure 1.2).

The read/write restrictions and authorizations on the global memory ac-

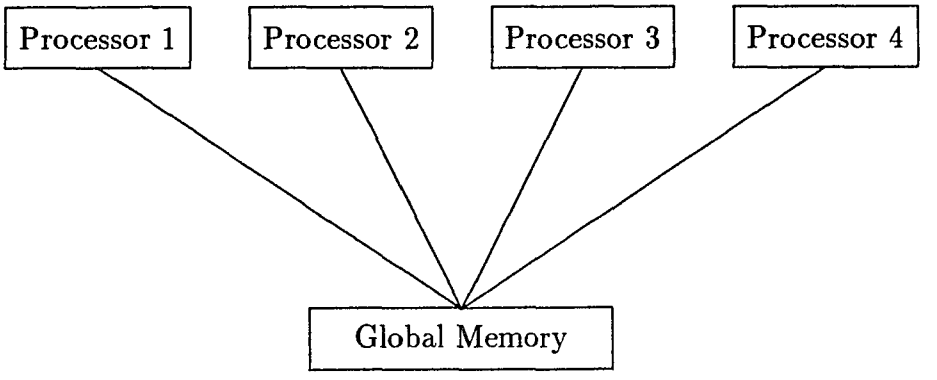



Figure 1.2: A 4-processor $PRAM$.

cess of a $PRAM$, which include the *exclusive-read* ($ER$) access, the *exclusive-write* ($EW$) access, the *concurrent-read* ($CR$) access and the *concurrent-write* ($CW$) access, have led to a classification of the $PRAM$ models into three classes: $EREW$, $CREW$, $CRCW$ (see [KR], [PP], [BP, chap. 4], [GL, page 276] or [Re93, page 13]). Note that the model $ERCW$ is plausible, but is not interesting and is omitted.

As has been pointed out in the passage from [KR] (see beginning of this section), the $PRAM$ models enable us to focus primarily on the concurrent aspects of the various operations required to solve a given problem, thus, leaving out the processor communication issues to each specific implementation. We will assume these models throughout this thesis (in particular, the derivation of the bounds on the computational complexity of our solutions will assume the $CREW$ $PRAM$ model), since they best accommodate our

high level description of the presented algorithms.

**Parallel computational complexity.** Under the *PRAM* models, the parallel complexity of an algorithm is measured by the number of parallel steps (time) and the corresponding number of required processors (equipment size) necessary to solve the relevant problem.

Hereafter, $t$ will denote a bound on the number of parallel steps required by a parallel algorithm to solve a given problem and $p$ will denote a bound on the corresponding number of required processors.

We will use the notation $(t,\ p) = O\,(t(n),\ p(n))$ to show that a parallel algorithm can run in $O(t(n))$ parallel steps by utilizing $O(p(n))$ processors.

**The B-principle (a variant of Brent's scheduling principle).** The number of processors required by the fastest parallel algorithm for a given problem is usually very large, but in practice, the number of processors available is fixed and limited. A simulation scheme, known as *Brent's scheduling principle*, is often utilized to slow down the fastest parallel algorithm for a given problem, so as to obtain a version of the same algorithm for the same problem capable of running with the available equipment.

We apply the Brent's scheduling principle in the following modified form, hereafter referred to as to the *B-principle* (see [P92a], [PP]), and expressed by the implication:

$$O(t, sp) \text{ implies } O(st, p), \text{ for any } s \geq 1.$$

This simulation also implies the following result:

THEOREM **1.4.1** ([Br74], see also [EG88]) or [KR]). *If a computation can be performed in $t$ parallel steps by using $q$ operations on any number of PRAM processors (of a given fixed type), then it can be performed in time $t+(q-t)/p$ by using $p$ processors (of the same type).*

PROOF. Suppose that at each step $i$, $1 \leq i \leq t$, the original computation performs $q_i$ operations, so that $q = \sum_{i=1}^{t} q_i$. Then step $i$ can be simulated on $p$ processors in time

$$\lceil q_i/p \rceil = \lfloor (q_i + p - 1)/p \rfloor \leq (q_i + p - 1)/p.$$

Consequently, if processor assignment is not a problem, the total simulation time

$$t' = \sum_{i=1}^{t} \lceil q_i/p \rceil = \sum_{i=1}^{t} \lfloor (q_i + p - 1)/p \rfloor \leq \sum_{i=1}^{t} (q_i + p - 1)/p = t + (q - t)/p. \quad \blacksquare$$

DEFINITION **1.4.1 (Randomized algorithms, $NC$ and $RNC$ classes).** An algorithm is *randomized* when it makes use of randomly generated data (see [Re93, page 16]). Some of the algorithms presented in this thesis utilize *randomization* of the *Las Vegas* type or *Monte Carlo* type. A randomized algorithm of the Las Vegas type always produces the correct output, if and when it terminates, but it may run indefinitely (our randomized algorithms for solving linear systems will have this characteristic). On the other hand, a randomized algorithm of the Monte Carlo type always terminates in some specified amount of time, but its output may give incorrect answer to the

relevant problem, with some specified probability (our randomized algorithms for computing the determinant will have this characteristic).

A computational problem $S$, with input length (size) $n$, is in $NC^k$ (respectively, randomized $NC^k$ or $RNC^k$), if it can be solved by a deterministic (respectively, randomized) algorithm in $O\left(\log^k n\right)$ parallel steps that use $n^{O(1)}$ processors. The $NC$ (respectively, randomized $NC$ or $RNC$) class (named so to honor Nicholas Pippenger) is defined as the set $NC = \bigcup_{k\geq 0}(NC^k)$ [respectively, the set $RNC = \bigcup_{k\geq 0}(RNC^k)$].

DEFINITION **1.4.2 (Potential work of a parallel algorithm,** see [BP] or [PP]). The product

$$\mathrm{W}(n) = t\,p = O(t(n)\,p(n))$$

is called the *potential work*. It represents the number of operations potentially executable, if all the $p$ processors, required by the relevant parallel algorithm to run in $t$ steps, were kept busy during the entire computation.

DEFINITION **1.4.3 (Efficiency and optimality,** see [KR]). One of the challenges for the designer of a parallel algorithm is to keep the potential work of his/her algorithm as close as possible to the running time of the fastest known sequential algorithm for solving the same problem, while also keeping his/her algorithm in $NC$. The concepts of *efficiency* and *optimality* of a parallel algorithm are defined based on these requirements. Denote polylog $(k) = \bigcup_{\alpha\geq 0} O(\log k)^\alpha$ and let $T(n)$ be the number of steps required by

the fastest known sequential algorithm for some computational problem. A parallel algorithm for the same problem is said to be processor (or work) *efficient* if its potential work W satisfies the equation $W(n)=O(T(n)\text{polylog}\,(n))$; it is said to be *optimal* if $W(n)=O(T(n))$. We may replace $T$ in the above definitions by a lower bound on the sequential complexity of the same computational problem. Note that efficiency (respectively, optimality) based on the latter definition implies efficiency (respectively, optimality) based on the former (see Chapter 6). In this thesis we only assume the former definition.

REMARK **1.4.1** (*NC* **versus polynomial speed-up algorithms).** The *NC* class is generally accepted as characterization of the problems that can be solved with a high degree of parallelism by using a feasible amount of hardware (see [Co81]). However, questions have been raised on the prescription to only focus on designing parallel algorithms that are efficient or optimum according to the above definitions. Some researchers have suggested more relaxed requirements based on the concept of *polynomial speed-up* which is defined as follows: if a problem $S$ requires sequential time $T(n)$, then a parallel algorithm which can solve the problem $S$ in time $t(n)$ is said to have *polynomial speed-up* if, for some $\epsilon < 1$, the relation $t(n) = O(T(n^{\epsilon}))$ is satisfied. These researchers argue that parallel algorithms with polynomial speed-up and a potential work $W(n) = O(T(n))$ (*constant inefficiency*) or $W(n) = O(T(n)\text{polylog}\,(n))$ (*polylog inefficiency*) may be extremely useful in practice, even though these algorithms do not run in polylog time (see

[KR]). These concepts have been successfully applied to solving a triangular linear systems and Toeplitz linear systems for which parallel algorithms having polynomial speed-up and constant inefficiency have been devised (see [PP]). Note that there are not any known $NC$ optimal parallel algorithms for solving these problems.

REMARK **1.4.2 (Efficiency, optimality and various *PRAM* models).** The complexity of matrix computations over the fields of characteristic zero does not depend on choosing any of the *EREW*, *CREW* and *CRCW* *PRAM* models. Over the finite fields and rings, the complexity may slightly vary [by a factor $O(\log n)$ for the time bound], depending on the choice of the model. A paper by Snir [Sn] shows that, in particular, the *CREW* model is strictly more powerful than the *EREW* model (see also [KR, page 31]). Consequently, over the finite fields and rings, it is necessary to indicate specific *PRAM* models, under which a fixed parallel algorithm is optimal, when such a claim is made, unless this algorithm is optimal under the weakest (*EREW*) *PRAM* model. The optimal algorithms presented in this thesis assume assume the *CREW PRAM* model.

REMARK **1.4.3 (*PRAM* versus circuits models).** If a problem can be solved by a *PRAM* algorithm using a number of processors polynomial in the size of the input and a time polynomial in the logarithm of the input size, it can be solved by a polynomial sized circuit with polylogarithmic depth,

and vice versa (see [EG88]). Consequently, if we had assumed the circuit model, our $NC$ algorithms would have remained in $NC$.

### 1.4.2 Some results in parallel computations.

Some methods have emerged as fundamental subroutines in the design of parallel algorithms. We recall some of these methods and results, the proof of which relies on the B-principle and can be found in [KR] or in [BP, chap. 4]. These results can be derived under any of the $PRAM$ models of parallel computations.

DEFINITION **1.4.4 (Prefix sums problem).** Let $\star$ denote an associative operation over a domain $\mathcal{D}$. Given an array $[x_1, \ldots, x_n]$ of $n$ elements from $\mathcal{D}$, the *prefix sum* problem is the computation of all the partial sums

$$S_i = x_1 \star x_2 \star \ldots \star x_i, \quad \text{for} \quad 1 \leq i \leq n. \tag{1.9}$$

PROPOSITION **1.4.1 (Parallel prefix computation,** [LF]). *The prefix sum problem can be solved, assuming any of the PRAM models, at an optimal and deterministic cost bounded by*

$$O\left(\log n, \frac{n}{\log n}\right). \tag{1.10}$$

PROOF. Assume (without loss of generality) that n is a power of 2 and perform the following computations:

ALGORITHM **1.4.1 (Parallel prefix algorithm).**

**Input:** *an array* $X = [x_1, \ldots, x_n]$, *where* $x_i \in (\mathcal{D}, \star)$.

**Output:** *an array* $S = [S_1, \ldots, S_n]$ *where* $S_i = \sum_{k=1}^{i} x_k$.

**Computations:** *if n=1, then set* $S_1 = x_1$; *else*

1. *concurrently compute the entries* $v_i$*'s of the auxiliary array denoted* $V = [v_1, \ldots, v_{n/2}]$, *where* $v_i = x_{2i-1} \star x_{2i}$;

2. *recursively apply "Parallel prefix algorithm" to* $V$. *Let* $Y = [y_1, \ldots, y_{n/2}]$ *denote the output of this recursive application;*

3. *concurrently compute* $S_1 = x_1$, $S_i = y_i$ *(if* $i$ *is even),* $S_i = y_{i-1} \star x_i$ *(if* $i$ *is odd), for* $2 \leq i \leq n$. ■

This algorithm computes all the required partial sums $S_i$. Moreover, the time bound $O(\log n)$ immediately follows (since steps 1 and 3 only take a constant time, and the recursive step 2 applies to a problem having half the size of the original problem). What does not appear obvious is the bound $O(n/\log n)$ on the number of processors required, since $O(n)$ processors seem to be necessary. Observe, however, that if we perform this algorithm assuming that only $O(n/\log n)$ processors are available, then only the last $\log\log n$ steps of the overall computation will actually be slowed down by a factor $\log n$. By applying the B-principle, it is easily checked that, in this case, this does not actually increase the overall asymptotic bound $O(\log n)$ on the number of steps required by the overall computation. ■

The B-principle also implies the following result, which generalizes the parallel prefix algorithm and which can be used to effectively combine several stages of a parallel algorithm:

PROPOSITION **1.4.2** [BP, chap. 4, prop. 1.1]. *If a computation consists of $S$ stages such that $p_s$ processors concurrently perform $t_s$ time-steps at stage $s$, for $s = 1, \ldots, S$, then this computation can be performed at a cost bounded by $O(t, p)$, where $t = \sum_{s=1}^{S} t_s$ and $p = \left\lceil \sum_{s=1}^{S} t_s p_s / t \right\rceil$.*

To conclude this section, we summarize the known parallel complexity estimates for some fundamental matrix computations.

PROPOSITION **1.4.3** ([BP], [P], [CW], [P91], [P92], [KP91], [KP92]). *Let $M(n, q, r)$, $I(q)$, $D(q)$ denote the problems of $n \times q$ by $q \times r$ matrix multiplication, $q \times q$ matrix inversion, and the evaluation of the determinant of a $q \times q$ matrix, respectively; then the parallel computational complexity of solving the above problems is given by*

$$\left(t_{M(n,q,r)},\, p_{M(n,q,r)}\right) = O\left(\log q,\, nqrh^{\omega-3}\right), \tag{1.11}$$

$$\left(t_{I(q)},\, p_{I(q)}\right) = O\left(\phi(\mathbf{F}, q),\, p_{M(q,q,q)}\right), \tag{1.12}$$

$$\left(t_{D(q)},\, p_{D(q)}\right) = O\left(\phi(\mathbf{F}, q),\, p_{M(q,q,q)}\right), \tag{1.13}$$

*respectively, where $h = \min\{n, q, r\}$, $2 \leq \omega < 2.376$, and*

$$\phi(\mathbf{F}, q) \leq \begin{cases} \log^2 q, & \text{if } \mathbf{F} \text{ has characteristic } 0, \\ \log^4 q, & \text{for any } \mathbf{F}, \end{cases}$$

*provided that the computation is performed over the field of constants $\mathbf{F}$.*

For practical purposes, one should assume that

$$\left(t_{M(n,q,r)},\ p_{M(n,q,r)}\right) = O\left(\log q,\ nqr/\log q\right), \tag{1.14}$$

due to the large overhead constant hidden in the "$O$" notation of (1.11). The bound (1.12) has been obtained in [P91], [P92], [KP91], [KP92], by using Las Vegas randomized algorithms.

## 1.5 Statement of the problems.

1. Our main focus in this thesis is the computational problem denoted $BAND \cdot LIN \cdot SOLVE = L \cdot S(n, w)$: given a nonsingular $n \times n$ matrix $A$ having bandwidth $w$, and an $n$-dimensional vector $\overrightarrow{\mathbf{b}}$, compute the vector $\overrightarrow{\mathbf{x}} = A^{-1}\overrightarrow{\mathbf{b}}$. We seek parallel solutions for this problem, which are in $NC^k$, for some constant $k$, to be made as small as possible. Moreover, the solution algorithm should be efficient or (preferably) optimal. Furthermore, the algorithm should provide a mechanism by which the computation of the solution to the problem $BAND \cdot LIN \cdot SOLVE \cdot SEV$, that is, to the problem of efficiently solving several linear systems of the format

$$A\overrightarrow{\mathbf{x}} = \overrightarrow{\mathbf{b}}(i) \quad (1 \leq i \leq s),$$

could be accelerated. By "efficiently solving" we mean solving so as to keep the potential work of solving the latter problem asymptotically

bounded by the potential work of the solution of $BAND{\cdot}LIN{\cdot}SOLVE$, for any $s \leq w$.

2. The nonsingularity assumption of $BAND \cdot LIN \cdot SOLVE$ can be verified by solving one of the two following problems (also of independent interest): $BAND{\cdot}DET{=}D(n,w)$: given an $n \times n$ matrix $A$, with bandwidth $w$, compute its determinant, $\det A$, or, if the computation is in the complex field or in one of its subfields, $|BAND \cdot DET|{=}|D(n,w)|^2$: given an $n \times n$ matrix $A$, with bandwidth $w$, compute $|\det A|^2$.

3. Besides the above two problems, we also seek algorithms for a special case of $BAND \cdot LIN \cdot SOLVE$, $BAND \cdot DET$ and $|BAND \cdot DET|$ denoted $BAND{\cdot}LIN{\cdot}SOLVE^{\star}$, $BAND{\cdot}DET^{\star}$ and $|BAND{\cdot}DET^{\star}|$, respectively, where the input matrix has lower and/or upper edge (see Definition 1.3.2, page 10). This requirement holds for a large class of banded matrices; in particular, it typically holds for the matrices encountered in applications to $PDE's$ and $ODE's$ (see [Am] and [LP]).

## 1.6 A sequential solution to a nonsingular BLS based on Gaussian elimination.

We first recall the classical Gaussian elimination applied to a banded linear system. The (sequential) solution to $BAND \cdot LIN \cdot SOLVE$ obtained via this technique, only requires

$$O(nw^2) \tag{1.15}$$

field operations, for any nonsingular linear system of $n$ equations having the bandwidth $w$. Moreover, when we solve several linear systems, each having the same coefficient matrix $A$, we may preprocess the matrix $A$ at a cost bounded by (1.15); after that, we may solve any linear system with the same coefficient matrix $A$, at a cost bounded by

$$O(nw). \tag{1.16}$$

This classical elimination technique reduces the original problem of solving a nonsingular linear system to solving a triangular linear system. This transformation is motivated by a relatively easy computation of the solution to a nonsingular triangular linear system $A\vec{x} = \vec{b}$, since the vector $\vec{x} = (x_1, \ldots, x_n)^T = A^{-1}\vec{b}$ can be obtained by utilizing either the *back substitution* formula (if $A$ is upper triangular)

$$x_i = \begin{cases} \frac{b_n}{a_{nn}}, & i = n, \\ \frac{b_i - \sum_{k=i+1}^{n} a_{ik}x_k}{a_{ii}}, & i = (n-1), \ldots, 1, \end{cases} \tag{1.17}$$

or the *forward substitution* formula (if $A$ is lower triangular)

$$x_i = \begin{cases} \frac{b_1}{a_{11}}, & i = 1, \\ \frac{b_i - \sum_{k=1}^{i-1} a_{ik}x_k}{a_{ii}}, & i = 2, \ldots, n. \end{cases} \tag{1.18}$$

The equations (1.17) and (1.18) imply that the solution of any nonsingular triangular linear system of $n$ equations only takes $O(n^2)$ field operations [compare with $O(n^3)$ required in the case of an arbitrary nonsingular linear

system]. Moreover, if the coefficient matrix has bandwidth $w$, then the back substitution formula (1.17) takes the form

$$x_i = \begin{cases} \frac{b_n}{a_{nn}}, & i = n; \\ \frac{b_i - \sum_{k=i+1}^{i+w} a_{ik}x_k}{a_{ii}}, & i = (n-1), \ldots, 1; \end{cases} \tag{1.19}$$

and the forward substitution formula (1.18) takes the form

$$x_i = \begin{cases} \frac{b_1}{a_{11}}, & i = 1, \\ \frac{b_i - \sum_{k=i-w}^{i-1} a_{ik}x_k}{a_{ii}}, & i = 2, \ldots, n. \end{cases} \tag{1.20}$$

The equations (1.19) and (1.20) immediately imply that any nonsingular triangular linear system of $n$ equations having bandwidth $w$ can be solved in sequential time bounded by $O(nw)$.

### 1.6.1 LU factorization.

A matrix $A$ has $LU$ factorization if it has a decomposition of the format

$$A = LU, \tag{1.21}$$

where $L$ is a lower triangular matrix and $U$ is an upper triangular matrix. If $A$ is nonsingular, has the bandwidth $w$ and if a decomposition of the format (1.21) has been computed, then both $L$ and $U$ are also nonsingular and, for any vector $\vec{\mathbf{b}}$, the vector $\vec{\mathbf{x}} = A^{-1}\vec{\mathbf{b}}$ is obtained by computing

$$\vec{\mathbf{y}} = L^{-1}\vec{\mathbf{b}}, \quad \text{and} \quad \vec{\mathbf{x}} = U^{-1}\vec{\mathbf{y}}; \tag{1.22}$$

moreover, it is easily checked that $w(L) = w_-(L) = w_-(A)$ and $w(U) = w_+(U) = w_+(A)$; consequently, the linear system $A\vec{\mathbf{x}} = \vec{\mathbf{b}}$ can be solved at a cost bounded by $O(nw)$ (provided that $L$ and $U$ are available). We will next show how to compute, for any nonsingular linear system $A\vec{\mathbf{x}} = \vec{\mathbf{b}}$, a matrix $U$ and a vector $\vec{\mathbf{y}}$ of (1.22), by means of *Gaussian elimination* [the method can be extended to compute $L$ and $U$ of (1.21), if such $L$ and $U$ exist (see [GL, page 97])].

## 1.6.2 Banded Gaussian elimination.

Let $A^{(1)} = (a_{ij}^{(1)})$ denote a nonsingular $n \times n$ matrix, $\vec{\mathbf{b}}^{(1)} = \left(b_1^{(1)}, \ldots, b_n^{(1)}\right)$ an $n$-dimensional vector and $S^{(1)}$ the linear system of equations

$$S^{(1)} : \begin{cases} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \ldots + a_{1n}^{(1)}x_n = b_1^{(1)} \\ a_{21}^{(1)}x_1 + a_{22}^{(1)}x_2 + \ldots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ \vdots \qquad\qquad\qquad\qquad\qquad\quad \vdots \\ a_{n1}^{(1)}x_1 + a_{n2}^{(1)}x_2 + \ldots + a_{nn}^{(1)}x_n = {}^{(1)}b_n. \end{cases} \tag{1.23}$$

Assuming that $a_{11}^{(1)} \neq 0$, perform the following computations:

ALGORITHM **1.6.1 (Gaussian elimination).**

**Input:** *a nonsingular $n \times n$ linear system $S^{(1)}$ of the format (1.23).*

**Output:** *an $n \times n$ linear system denoted $S^{(2)}$ with coefficient matrix $A^{(2)} = (a_{ij}^{(2)})$, equivalent to $S^{(1)}$, and such that $a_{i1}^{(2)} = 0$, for all $i \geq 2$ [see (1.24) below].*

**Computations:**

1. *let $\lambda(1)$ denote the first equation of $S^{(1)}$;*

2. *for* $2 \leq k \leq n$, *multiply both sides of* $\lambda(1)$ *by* $(-a_{k1}^{(1)}/a_{11}^{(1)})$, *then add the resulting equation to the* $k$*th equation of* $S^{(1)}$ *to form a new equation, denoted* $\lambda(k)$;

3. *output* $S^{(2)} = \{\lambda(i)\}_{1 \leq i \leq n}$; ■

The linear system $S^{(2)}$, output by the above algorithm, has the format

$$S^{(2)}: \begin{cases} a_{11}^{(2)}x_1 + a_{12}^{(2)}x_2 + \ldots + a_{1n}^{(2)}x_n = b_1^{(2)} \\ a_{22}^{(2)}x_2 + \ldots + a_{2n}^{(2)}x_n = b_2^{(2)} \\ \vdots \qquad\qquad\qquad\qquad \vdots \\ a_{n2}^{(2)}x_2 + \ldots + a_{nn}^{(2)}x_n = b_n^{(2)}; \end{cases} \tag{1.24}$$

where

$$a_{kj}^{(2)} = \begin{cases} a_{1j}^{(1)}, & k = 1; \\ a_{kj}^{(1)} - \frac{a_{k1}^{(1)}}{a_{11}^{(1)}} a_{1j}^{(1)}, & k = 2, \ldots, n; \end{cases}$$

$$b_k^{(2)} = \begin{cases} b_1^{(1)}, & k = 1; \\ b_k^{(1)} - \frac{a_{k1}^{(1)}}{a_{11}^{(1)}} b_1^{(1)}, & k = 2, \ldots, n. \end{cases}$$

If $a_{22}^{(2)} \neq 0$, we may apply Algorithm 1.6.1 (see page 27) to the subsystem of $S^{(2)}$ formed by its last $(n-1)$ equations [note that the last $(n-1)$ equations of $S^{(2)}$ are free of the variable $x_1$]. We will assume (without loss of generality) that the coefficients $a_{kk}^{(k)}$ of all the auxiliary linear systems $S^{(k)}$ satisfy the condition

$$a_{kk}^{(k)} \neq 0, \text{ for } 1 \leq k < n. \tag{1.25}$$

Over the fields of characteristic zero, the condition (1.25) can be ensured by symmetrization, or, alternatively, over any field, by *partial pivoting* (see

§1.6.3). Apply the above technique $n-1$ times to arrive at the upper triangular linear system denoted

$$S^{(n)}: \ A^{(n)}\vec{\mathbf{x}} = \vec{\mathbf{b}}^{(n)}$$

[note that the matrix $A^{(n)}$ and the vector $\vec{\mathbf{b}}^{(n)}$ correspond to $U$ and $\vec{\mathbf{y}}$ of (1.22), respectively, for the linear system $A\vec{\mathbf{x}} = \vec{\mathbf{b}}$]. It is easily checked that the transition from $S^{(1)}$ to $S^{(n)}$ takes $O(n^3)$ field operations, for any $n \times n$ nonsingular input linear system. However, if the matrix $A$ has bandwidth $w$, then $O(nw^2)$ operations will suffice.

The Gaussian elimination technique is immediately extended to accelerate the solution of $q$ linear systems, each having the same $n \times n$ banded coefficient matrix $A$, at a cost bounded by $O(nw^2+nwq)$, where $w = w(A)$. This bound can be obtained by replacing, in Algorithm 1.6.1, the vectors $\vec{\mathbf{x}}$ and $\vec{\mathbf{b}}$ by two $n \times q$ matrices denoted $X$ and $B$, respectively. Alternatively, we may compute the matrix $A^{(n)}$, in a preprocessing stage, at a cost bounded by $O(nw^2)$, then, compute the solution of any $q$ linear systems, each having coefficient matrix $A$, at a cost bounded by $O(nwq)$.

### 1.6.3 Pivoting.

The coefficient $a_{kk}^{(k)}$ $(1 \leq k \leq n)$ (the diagonal entries of the upper triangular matrix $A^{(n)}$ output by Gaussian elimination) are called *pivoting elements.* If some pivoting element (say, $a_{kk}^{(k)}$) happens to be zero, we still may carry out the $k$th stage in Gaussian elimination at the price of interchanging the

$k$th equation with some $(k+l)$th equation of the linear system $S^{(k)}$, where $l > 0$ and $a_{k+l,k}^{(k)} \neq 0$. If the original input matrix $A$ is nonsingular, then it is guaranteed that such an $l$ can be found, otherwise the rank of the matrix formed by the southeastern $(n-k+1) \times (n-k+1)$ submatrix of the matrix $A^{(k)}$ would be less than $(n-k+1)$, which is not possible. This row swapping technique, known as *partial pivoting*[5], only requires that at most $n-k$ entries be searched (if the input matrix has the lower bandwidth $w_-$, then only $w_-$ entries need be searched). Moreover, the row interchange only takes $O(1)$ operations (for demonstration, if each matrix is represented by a 2-dimensional array, we may simply swap the pointers used to identify the relevant rows). The total work required for both searching and swapping does not increase the asymptotic computational complexity of the $k$th step of Gaussian elimination. Note that the bandwidth of the coefficient matrix of the linear system resulting from a row interchange does not exceed $2w(A)$ (due to the fact that the column of the current pivot element contains some nonzero coefficients in the $w_-(A)$ rows below this current pivoting element). Consequently, the bounds (1.15) and (1.16) remain valid, even when pivoting is required in Gaussian elimination.

REMARK **1.6.1** If the input matrix has size $n \times n$, then the Gaussian elimination algorithm requires $n-1$ successive steps, and the computations at each

[5]Another swapping strategy called *complete pivoting* leads to a numerically more stable algorithm, although in practice of numerical computations, partial pivoting usually suffices (see [DB, page 150]).

step depends on the results of the previous step. Because of these dependencies, parallel implementation of this technique leads to parallel algorithms that solve a linear system in times at least $n - 1$.

REMARK **1.6.2 Block band Gaussian elimination.** We may apply the Gaussian elimination technique to a block matrix. In this case, nonsingularity of all the pivoting blocks is required and can be ensured via symmetrization in the case of computation over the fields of characteristic zero. Over any field, we may ensure nonsingularity of each pivoting block via randomization. Specifically, for any $q \times q$ block tridiagonal nonsingular linear system with $w \times w$ blocks, we may premultiply both sides of the auxiliary linear system output at the $k$-th stage of Gaussian elimination by a matrix of the format $P = \text{diag}\,(I_1, R, I_2)$, where $I_1$ and $I_2$ are the identity matrices of appropriate sizes and $R$ is a $(2w) \times (2w)$ random matrix. It is immediately verified that the bandwidth of any of the auxiliary matrices remains bounded by $2w$. Since there is an assignment of the entries of $R$ for which the current block pivot is nonsingular, nonsingularity of this pivoting elements follows, with a high probability, for a random assignment from a large fixed set to the entries of $R$. The computational complexity of this block banded Gaussian elimination is bounded by $O(qt_{I(w)}, p_{I(w)})$.

## 1.7 The main results.

Recall that our main goal is to develop techniques leading to parallel algorithms for solving the problems $BAND \cdot LIN \cdot SOLVE$ and $BAND \cdot DET$, which are in $NC^k$ or $RNC^k$, for some constant $k$. Of course, we wish to make $k$ as small as possible, and, moreover, the algorithms should be work efficient or (preferably) optimal. A recent work of W. Eberly [E] (see our chapter 3), which, until the appearance of [PSA], held the record bounds on the computational complexity of these problems, was a major step towards this goal.

By following the techniques of [PSA], we show a substantial improvement of the estimates of [E] [see (3.1), (3.2), page 54]. In particular, we reach the Las Vegas randomized (respectively, deterministic) bound

$$O\left(\left(\log \frac{n}{w}\right) t_{I(w)}, \left(\frac{n}{w}\right) p_{I(w)}\right), \tag{1.26}$$

for the problem $BAND \cdot LIN \cdot SOLVE$ over any field (respectively, any field of characteristic zero). However, in the fields of characteristic zero, it is preferable to solve this problem by using the block cyclic reduction algorithm, which reaches, for the same problem, the *optimal and deterministic* computational cost bound

$$O\left(\left(\log \frac{n}{w}\right) t_{I(w)}, \frac{\left(\frac{n}{w}\right) p_{I(w)}}{\left(\log \frac{n}{w}\right)}\right). \tag{1.27}$$

Moreover, regarding the problem $BAND \cdot DET$, we reach the optimal

randomized bound

$$O\left(\left(\log\frac{n}{w}\right)\left(t_{I(w)}+t_{D(w)}\right),\ \frac{\left(\frac{n}{w}\right)\left(p_{I(w)}+t_{D(w)}\right)}{\left(\log\frac{n}{w}\right)}\right). \tag{1.28}$$

over *any* field of constants; over the fields of characteristic zero, the problem $|BAND \cdot DET|$ can be solved deterministically at a cost bounded by (1.28).

Furthermore, regarding the problems $BAND{\cdot}LIN{\cdot}SOLVE^{\star}$ and $BAND{\cdot}DET^{\star}$ (which denote the problems $BAND{\cdot}LIN{\cdot}SOLVE$ and $BAND{\cdot}DET$, respectively, in the case where the input matrix $A$ has lower edge and/or an upper edge, see Definition 1.3.2, page 10), we reach, *over any field of constants*, the improved deterministic and optimal computational cost bounds

$$O\left(\left(\log\frac{n}{w}\right)(\log w)+t_{I(w)},\ \frac{\left(\frac{n}{w}\right)t_{I(w)}\,p_{I(w)}}{t_{I(w)}+\left(\log\frac{n}{w}\right)(\log w)}\right) \tag{1.29}$$

and

$$O\left(\left(\log\frac{n}{w}\right)(\log w)+t_{I(w)}+t_{D(w)},\ \frac{\left(\frac{n}{w}\right)\left(t_{I(w)}\,p_{I(w)}+t_{D(w)}\,p_{D(w)}\right)}{t_{I(w)}+t_{D(w)}+\left(\log\frac{n}{w}\right)(\log w)}\right), \tag{1.30}$$

respectively.

Each of the solutions presented for the problem $BAND \cdot LIN \cdot SOLVE$ can be extended to accelerate the solution of several linear systems, each having the same input matrix $A$, so as to satisfy the requirements listed in §1.5 for the problem $BAND{\cdot}LIN{\cdot}SOLVE{\cdot}SEV$. To achieve this extension, we will divide each of our algorithms into two stages:

1. A *preprocessing* stage, $PREPROCESS{=}P(n,w)$: given a nonsingular $n \times n$ matrix $A$ having bandwidth $w$, compute a set $\Gamma(A)$ of parameters,

implicitly defining the inverse matrix $A^{-1}$. A choice for the set $\Gamma(A)$ depends on each specific problem, as well as on each solution algorithm. Various choices for $\Gamma(A)$ will be specified in chapters 2–5.

2. A *backsolving* stage, denoted $BACK \cdot SOLVE{=}B(n,w)$: given a non-singular $n \times n$ matrix $A$ with bandwidth $w$, a vector $\overrightarrow{\mathbf{b}}$, and the set $\Gamma(A)$ of parameters output by preprocessing, compute the vector $A^{-1}\overrightarrow{\mathbf{b}}$.

To solve $s$ linear systems, in the case of any fields, we first compute the set $\Gamma(A)$ at a cost bounded by (1.26); after that, we *deterministically* compute the solution to $s$ linear systems, each having coefficient matrix $A$, at an efficient cost bounded by

$$O\left(\left(\log\frac{n}{w}\right)(\log w), \frac{wns}{\log w}\right). \tag{1.31}$$

Over the fields of characteristic zero, we compute the set $\Gamma(A)$ at a deterministic and optimal cost bounded by (1.27), by applying the block cyclic reduction technique; after that, we *deterministically* compute the solution to $s$ linear systems, each having coefficient matrix $A$, at an optimal cost bounded by

$$O\left(\left(\log\frac{n}{w}\right)(\log w), \frac{wns}{\left(\log\frac{n}{w}\right)(\log w)}\right). \tag{1.32}$$

In the case where the matrix $A$ has a lower and/or an upper edge, we compute the set $\Gamma(A)$ at a deterministic computational cost bounded by (1.29); then, we *deterministically* compute the solution to $s$ linear systems,

each having coefficient matrix $A$, at a cost bounded by

$$O\left(\left(\log \frac{n}{w}\right)(\log w),\ \frac{wns}{\left(\log \frac{n}{w}\right)(\log w)}\right). \tag{1.33}$$

The bound (1.33) also applies when the input matrix is block bidiagonal (which includes the banded triangular case).

Let us summarize: By using the techniques of [PSA], we show substantial improvements of the previous record bounds of [E]. These algorithms, as well as the algorithms of [E], are efficient, but, unlike [E], we reach the $RNC^1$ time bound $O(\log n)$ (if $w$ is a constant). Moreover, we reach an optimum bound for the problem $BAND{\cdot}DET$ over any field of constants; furthermore, we arrived at the improved, optimal and deterministic complexity estimates for $BAND \cdot LIN \cdot SOLVE$ when the input matrix is either block bidiagonal or block tridiagonal with nonsingular subdiagonal (or superdiagonal) blocks. Finally, we accelerated the computation of the solution to several linear systems, each having the same banded coefficient matrix, by utilizing preprocessing algorithms.

# Chapter 2

## An optimal parallel solution for a BLS over the fields of characteristic zero, based on block cyclic reduction.

*Block cyclic reduction* (BCR) is a divide and conquer technique, frequently utilized for solving special BLS's, which arise from various engineering applications (see [Ho], [GL, page 173], [BD], [BGN], [He], [Sw] [Swe74], [Swe77]). The main idea (which is due to Hockney [Ho]) is to recursively reduce the input problem to a similar problem of half size (by eliminating half of the variables), until we are left with a linear system of a sufficiently small size, which is then solved by some standard means; the previously eliminated variables are then computed via back substitution.

In this chapter we apply the BCR technique to solving $BAND \cdot LIN \cdot SOLVE$ (the problem of solving any nonsingular banded linear system) over the fields of characteristic zero. This solution of $BAND \cdot LIN \cdot SOLVE$

supports the *deterministic* bound

$$(t_{bcr}(n,w), p_{bcr}(n,w)) = O\left(\left(\log \frac{n}{w}\right) t_{I(w)}, \left(\frac{\frac{n}{w}}{\log \frac{n}{w}}\right) p_{I(w)}\right), \tag{2.1}$$

assuming any $n \times n$ nonsingular input matrix $A$ having bandwidth $w$.

The bound (2.1) implies that the BCR solution of $BAND \cdot LIN \cdot SOLVE$ is in $NC$ (and even in $NC^1$, if $w$ is a constant) and is *optimum* according to the definition of [KR], since its *potential work* $\mathrm{W}_{bcr}(n,w) = t_{bcr}(n,w) \times p_{bcr}(n,w)$ supports the bound $O(nw^2)$ [which is the record sequential bound on the cost of solving the problem $BAND \cdot LIN \cdot SOLVE$, see §1.6, page 24].

Moreover, we extend the BCR algorithm to accelerate the computation of the solution to several linear systems each having the same banded coefficient matrix. To achieve this acceleration, we first preprocess the matrix $A$ at a cost bounded by (2.1); after that, we solve any linear system having coefficient matrix $A$, at an optimum *deterministic* cost bounded by

$$O\left(\left(\log \frac{n}{w}\right)(\log w), \frac{wn}{\left(\log \frac{n}{w}\right)(\log w)}\right), \tag{2.2}$$

so that we solve $s$ linear systems, each having the same coefficient matrix $A$, at an optimum deterministic cost bounded by

$$O\left(\left(\log \frac{n}{w}\right)(\log w), \frac{swn}{(\log \frac{n}{w})(\log w)}\right). \tag{2.3}$$

The preprocessing algorithm computes a set of parameters denoted $\Gamma_{bcr}(A)$, implicitly defining the matrix $A^{-1}$.

Furthermore, we extend the BCR algorithm to the evaluation of $|\det A|$, for any $n \times n$ matrix $A$ having bandwidth $w$, at a deterministic cost bounded by

$$O\left(\left(\log \frac{n}{w}\right)\left(t_{D(w)} + t_{I(w)}\right), \frac{\left(\frac{n}{w}\right)}{\log\left(\frac{n}{w}\right)}\left(p_{D(w)} + p_{I(w)}\right)\right). \tag{2.4}$$

The algorithm for computing $|\det A|$ does not assume nonsingularity of $A$; therefore, it can be utilized to test if $A$ is singular.

Regarding the overall storage space of the solution to several linear systems, the preprocessing stage requires some extra storage, besides the storage needed for the input linear system(s). However, this extra space is no more than twice that required for the input matrix.

We will utilize the notation $q$-$w$-block-3DM (respectively, $q$-$w$-block-3DLS) for any $q \times q$ block tridiagonal matrix (respectively, $q \times q$ block tridiagonal linear system) where each matrix block has size $w \times w$. We will assume a nonsingular $q$-$w$-block-3DM input matrix; all the results are immediately applied to any nonsingular $n \times n$ banded linear system having bandwidth $w$; this can be done by setting $q = n/w$ [see (1.6), page 11].

We organize this chapter in the following order: In §2.1, we recall the classical BCR algorithm applied to an h.p.d. $q$-$w$-block-3DLS. In §2.2, we preprocess a $q$-$w$-block-3DM, based on block cyclic reduction, and we apply this preprocessing technique to accelerate the computation of the solution of several $q$-$w$-block-3DLS's, each having the same h.p.d. coefficient matrix; In §2.3, we relax the h.p.d. assumption. In §2.4 we extend the BCR algorithm

to the computation of $|\det A|$.

## 2.1 Block cyclic reduction applied to an h.p.d. block tridiagonal linear system.

Hereafter, $A = A^{(r)}$ denotes a $q$-$w$-block-3DM of the format

$$A = A^{(r)} = \begin{pmatrix} D_1 & F_1 & & & & & \\ E_2 & D_2 & F_2 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & E_{q-1} & D_{q-1} & F_{q-1} \\ & & & & & E_q & D_q \end{pmatrix}, \tag{2.5}$$

where we assume (with no loss of generality) that

$$q = 2^r - 1, \quad r \geq 1. \tag{2.6}$$

Our input linear system ($A\vec{\mathbf{x}} = \vec{\mathbf{b}}$) will be denoted $S^{(r)}$. We will also consider the following alternative representation of $S^{(r)}$:

$$E_i \vec{\mathbf{x}}_{i-1} + D_i \vec{\mathbf{x}}_i + F_i \vec{\mathbf{x}}_{i+1} = \vec{\mathbf{b}}_i, \quad 1 \leq i \leq q, \tag{2.7}$$

where $\vec{\mathbf{x}} = \left(\vec{\mathbf{x}}_1, \ldots, \vec{\mathbf{x}}_q\right)^T$, $\vec{\mathbf{b}} = \left(\vec{\mathbf{b}}_1, \ldots, \vec{\mathbf{b}}_q\right)^T$, $\vec{\mathbf{b}}_k$ and $\vec{\mathbf{x}}_k$ are $w$-dimensional column vectors, $E_1 = F_q = 0_w$, $\vec{\mathbf{x}}_0 = \vec{\mathbf{x}}_{q+1} = \vec{\mathbf{0}}$.

We seek a smaller linear system denoted $S^{(r-1)}$ whose solution partially solves $S^{(r)}$. Until §2.3, we will assume that $A$ is an h.p.d. matrix. Under this condition, we may apply the following algorithm to $S^{(r)}$, in order to compute such a smaller linear system:

ALGORITHM **2.1.1 (BCR-reduce).**

**Input:** *an h.p.d. q-w-block-3DLS of the format (2.7), denoted* $S^{(r)}$.

**Output:** *a* $q'$*-w-block-3DLS* ($q' = \frac{q-1}{2} = 2^{r-1} - 1$) *of the format (2.7) as well [denoted* $S^{(r-1)}$*] whose solution determines the values of the even-indexed variables of* $S^{(r)}$.

**Computations:**

**for** $j = 1$ **to** $q'$ **do** (in parallel for all values of $j$)

1. *substitute* $2j-1$ *for* $i$ *in (2.7) and premultiply both sides of the resulting equation by the matrix* $(-E_{2j}D_{2j-1}^{-1})$ *to form a new equation denoted* $\lambda_1(j)$;
2. *substitute* $2j$ *for* $i$ *in (2.7) to form a new equation denoted* $\lambda_2(j)$;
3. *substitute* $2j+1$ *for* $i$ *in (2.7) and premultiply both sides of the resulting equation by the matrix* $(-F_{2j}D_{2j+1}^{-1})$ *to form a new equation denoted* $\lambda_3(j)$;
4. *add the three equations* $\lambda_1(j)$, $\lambda_2(j)$ *and* $\lambda_3(j)$ *to form the output equations* $\lambda(j)$, $1 \le j \le q'$. ■

The linear system $S^{(r-1)}$, output by the above algorithm on input $S^{(r)}$, has the format

$$E_j^{(r-1)}\overrightarrow{\mathbf{x}}_{2j-2} + D_j^{(r-1)}\overrightarrow{\mathbf{x}}_{2j} + F_j^{(r-1)}\overrightarrow{\mathbf{x}}_{2j+2} = \overrightarrow{\mathbf{b}}_j^{(r-1)}, \quad 1 \le j \le q', \tag{2.8}$$

where $q' = 2^{r-1} - 1$ and

$$E_j^{(r-1)} = -E_{2j}D_{2j-1}^{-1}E_{2j-1}, \tag{2.9}$$

$$D_j^{(r-1)} = -E_{2j}D_{2j-1}^{-1}F_{2j-1} + D_{2j} - F_{2j}D_{2j+1}^{-1}E_{2j+1}, \tag{2.10}$$

$$F_j^{(r-1)} = -F_{2j}D_{2j+1}^{-1}F_{2j+1}, \tag{2.11}$$

$$\overrightarrow{\mathbf{b}}_j^{(r-1)} = -E_{2j}D_{2j-1}^{-1}\overrightarrow{\mathbf{b}}_{2j-1} + \overrightarrow{\mathbf{b}}_{2j} - F_{2j}D_{2j+1}^{-1}\overrightarrow{\mathbf{b}}_{2j+1}. \tag{2.12}$$

A simple inspection shows that $S^{(r-1)}$ is a $q'$-$w$-block-3DLS. Moreover, $S^{(r-1)}$ is Hermitian; to verify this assertion, observe that $(E_{k+1})^H = F_k$ (since $S^{(r)}$ is Hermitian); consequently,

$$\begin{aligned}
\left(E_{j+1}^{(r-1)}\right)^H &= -\left(E_{2(j+1)}D_{2j+1}^{-1}E_{2j+1}\right)^H \\
&= -\left(E_{2j+1}\right)^H \left(D_{2j+1}^{-1}\right)^H \left(E_{2(j+1)}\right)^H \\
&= -F_{2j}D_{2j+1}^{-1}F_{2j+1} \\
&= F_j^{(r-1)}.
\end{aligned}$$

Furthermore, $S^{(r-1)}$ is positive definite (see [He, Remark 7]). We conclude that $S^{(r-1)}$ is h.p.d., so that Algorithm 2.1.1 (BCR-reduce) can be applied (recursively) to $S^{(r-1)}$. We may now summarize the BCR algorithm:

ALGORITHM **2.1.2 (Block cyclic reduction).**

**Input:** *an h.p.d. q-w-block-3DLS* ($A\overrightarrow{\mathbf{x}} = \overrightarrow{\mathbf{b}}$) *of the format (2.7), denoted* $S^{(r)}$, *where* $q = 2^r - 1$, $r \geq 1$.

**Output:** $\overrightarrow{\mathbf{x}} = \left(\overrightarrow{\mathbf{x}}_1^T, \ldots, \overrightarrow{\mathbf{x}}_q^T\right)^T = A^{-1}\overrightarrow{\mathbf{b}}$.

**Computations:**

**if** $r = 1$ **then** *solve* $S^{(r)}$ *by using available routines;*

**else**

1. *Let* $S^{(r-1)}$ *denote the linear system computed by applying Algorithm 2.1.1 (BCR-reduce) to* $S^{(r)}$*;*
2. *solve* $S^{(r-1)}$ *by (recursively) applying this block cyclic reduction algorithm to* $S^{(r-1)}$ *[note that after this step, the values of the even-indexed variables of* $S^{(r)}$ *are available from the solution of* $S^{(r-1)}$*];*
3. *concurrently compute the odd-indexed variables of* $S^{(r)}$ *via back substitution.* ■

EXAMPLE **2.1.1** For demonstration, let $r = 2$, $q = 3 = 2^r - 1$; then, the linear system $S^{(2)}$ consists of three vector equations of the format

$$\begin{cases} D_1\vec{\mathbf{x}}_1 + F_1\vec{\mathbf{x}}_2 = \vec{\mathbf{b}}_2, \\ E_2\vec{\mathbf{x}}_1 + D_2\vec{\mathbf{x}}_2 + F_2\vec{\mathbf{x}}_3 = \vec{\mathbf{b}}_2, \\ E_3\vec{\mathbf{x}}_2 + D_3\vec{\mathbf{x}}_3 = \vec{\mathbf{b}}_3. \end{cases} \tag{2.13}$$

When we apply Algorithm 2.1.1 (BCR-reduce) to (2.13), the output $S^{(1)}$ is a linear system consisting of a single vector equation

$$(-E_2D_1^{-1}F_1 + D_2 - F_2D_3^{-1}E_3)\vec{\mathbf{x}}_2 = -E_2D_1^{-1}\vec{\mathbf{b}}_1 + \vec{\mathbf{b}}_2 - F_2D_3^{-1}\vec{\mathbf{b}}_3. \tag{2.14}$$

We solve this small linear system by directly computing the inverse of its coefficient matrix; after that, the value (say, $\vec{v}_2$) of $\vec{x}_2$ is available; the vectors $\vec{x}_1$ and $\vec{x}_3$ are concurrently computed by simultaneously substituting $\vec{v}_2$ for $\vec{x}_2$ in the first and third equations of (2.13), respectively.

Let us now estimate the parallel computational complexity of block cyclic reduction algorithm:

THEOREM **2.1.1** *The computational cost of the block cyclic reduction (Algorithm 2.1.2) applied to solving any h.p.d. $q \times q$ block tridiagonal linear system, with $w \times w$ blocks, is bounded by*

$$O\left((\log q)t_{I(w)},\ \frac{q}{\log q}\, p_{I(w)}\right), \tag{2.15}$$

*which turns into*

$$O\left(\left(\log \frac{n}{w}\right) t_{I(w)},\ \frac{\frac{n}{w}}{\log \frac{n}{w}}\, p_{I(w)}\right), \tag{2.16}$$

*for $q = n/w$.*

PROOF. Stage 1 of the block cyclic reduction algorithm (see page 41) [that is, the computation of $S^{(r-1)}$ given $S^{(r)}$ ] can be performed at the cost $O(t_{I(w)},\ (q/2)\, p_{I(w)})$. Its stage 2 solves the smaller linear system $S^{(r-1)}$ via a recursive application of this algorithm to $S^{(r-1)}$. Once $S^{(r-1)}$ is solved, the even-indexed variables of $S^{(r)}$ are known and stage 3 concurrently computes the odd-indexed variables of $S^{(r)}$; the value of each of these odd-indexed variables (say, $\vec{x}_{2h-1}$) is computed by substituting $(2h-1)$ for $i$ in the equation

(2.7) and by replacing the vectors $\vec{\mathbf{x}}_{2h}$ and $\vec{\mathbf{x}}_{2h+2}$ in the resulting equation by their values available from the solution of $S^{(r-1)}$; in this way, the vector $\vec{\mathbf{x}}_{2h-1}$ is effectively computed at the cost of a constant number of $w \times w$ matrix by a $w$-dimensional vector multiplications. The computational cost of each step in the back substitution process is dominated by the cost of the computations required in the corresponding reduction step; consequently, in order to estimate the overall asymptotic computational cost of the BCR algorithm, it is sufficient to estimate the cost of all the reduction steps. Toward this goal, let $t = t(r,w)$ denote the number of parallel steps required for computing the solution of $S^{(r)}$ and let $p = p(r,w)$ denote the corresponding number of required processors. The above discussion leads to the following relations for the pair $(t,p)$, estimated within a multiplicative constant factor:

$$t(r,w) \leq \begin{cases} t_{I(w)}, & r = 1; \\ t(r-1,w) + t_{I(w)}, & \text{otherwise;} \end{cases}$$

$$p(r,w) \leq \begin{cases} p_{I(w)}, & r = 1; \\ \max\left\{p(r-1,w),\ 2^{r-1}p_{I(w)},\right\}, & \text{otherwise.} \end{cases}$$

Recursive application of the latter relations and the B-principle yields the desired complexity bound (2.15). ∎

## 2.2 Preprocessing and solution of an h.p.d. block tridiagonal linear system, based on block cyclic reduction.

We will assume in this section the notation of the previous section; in addition, hereafter, $A^{(h)}$ denotes the coefficient matrix of the linear system $S^{(h)}$ and $E_i^{(h)}$, $D_j^{(h)}$, $F_k^{(h)}$ denote the blocks of $A^{(h)}$, for an appropriate choice of the indices $h$, $i$, $j$ and $k$.

We define our set $\Gamma(A) = \Gamma(A^{(r)})$ (the set of parameters computed by the preprocessing algorithm) by the relation

$$\Gamma_{bcr}(A^{(r)}) = \begin{cases} \left\{\left(A^{(r)}\right)^{-1}\right\}, & r = 1; \\ \gamma_{bcr}(A^r) \cup \left\{A^{(r-1)}\right\} \cup \Gamma_{bcr}\left(A^{(r-1)}\right), & r > 1; \end{cases} \tag{2.17}$$

$$\gamma_{bcr}(A^{(r)}) = \left\{\left(D_1^{(r)}\right)^{-1}, \left(D_3^{(r)}\right)^{-1}, \ldots, \left(D_q^{(r)}\right)^{-1}\right\}.$$

Note that the set $\Gamma_{bcr}(A)$ includes all the matrices $A^{(h)}$ $(1 \leq h < r)$ and the inverses of all the odd-indexed diagonal blocks of the matrices $A^{(k)}$ $(1 \leq k \leq r)$.

THEOREM **2.2.1**

1. *The cost of computing the set $\Gamma_{bcr}(A)$, for any h.p.d. $q \times q$ block tridiagonal matrix $A$ with $w \times w$ blocks is bounded by (2.15).*

2. *Once the set $\Gamma_{bcr}(A)$ has been computed, the vector $A^{-1}\vec{\mathbf{b}}$ can be computed at a cost bounded by*

$$O\left((\log q)(\log w), \frac{qw^2}{(\log q)(\log w)}\right), \tag{2.18}$$

*which turns into*

$$O\left(\left(\log\frac{n}{w}\right)(\log w), \frac{nw}{\left(\log\frac{n}{w}\right)(\log w)}\right), \tag{2.19}$$

*for* $q = n/w$.

PROOF. Observe that once the set $\Gamma_{bcr}(A)$ is available, the left hand sides of the equations in any of the linear systems $S^{(h)}$ are known. The computation of the right hand sides of these equations as well as the subsequent back substitution only require $w \times w$ matrix-by-vectors multiplications (note that no $w \times w$ matrix-by-matrix multiplication is performed). The proof of Theorem 2.2.1 immediately follows from these observations. ∎

REMARK **2.2.1** The equation (2.17) implies that the storage space required for the set $\Gamma_{bcr}(A)$ is no more than twice the storage space of the original input matrix $A$.

## 2.3 Relaxing the h.p.d. assumption.

Shifting from $A$ to $A^HA$ enables us to relax the h.p.d. assumption about the coefficient matrix of the input linear system. This is a valid transition, due to Propositions 1.2.1, 1.3.1 (see pages 8, 10) and the equation $A^{-1} = (A^HA)^{-1}A^H$ which enable us to reduce the computation of the vector $A^{-1}\overrightarrow{\mathbf{b}}$ (for any nonsingular $n \times n$ matrix $A$ and any $n$-dimensional vector $\overrightarrow{\mathbf{b}}$) to the computation of the vector $A'^{-1}\overrightarrow{\mathbf{b}'}$, where $A' = (A^HA)$ and $\overrightarrow{\mathbf{b}'} = A^H\overrightarrow{\mathbf{b}}$. This reduction amounts to a single multiplication of an $n \times w$ matrix by a vector

of dimension $w$, plus a single multiplication of matrices of size $n \times w$. If the matrix $A$ is nonsingular, then the matrix $A'$ is h.p.d.; consequently, the BCR algorithm solves the problem $BAND \cdot LIN \cdot SOLVE$ at a computational cost bounded by (2.1); this can be done by simply computing the vector $A'^{-1}\overrightarrow{\mathbf{b}'}$.

REMARK **2.3.1** The extension of the block cyclic reduction algorithm to the case of any field of constants does not appear obvious. The block cyclic reduction algorithm (see page 41) requires nonsingularity of all the odd-indexed diagonal blocks of the matrices $A^{(h)}$, $1 \leq h \leq r-1$. These requirements are satisfied if the input matrix is h.p.d.; this restriction is not a major handicap, provided that the ground field has characteristic zero, since it can be relaxed via symmetrization. But the notion of h.p.d. matrix is not defined for matrices over arbitrary fields. If the BCR algorithm could be applied to all strongly nonsingular linear systems, we could perhaps find some ways for ensuring required nonsingularity (of the odd-indexed diagonal blocks of $A$). Unfortunately, as the following simple example shows, Algorithm 2.1.2 may fail when the input matrix is strongly nonsingular, but is not h.p.d.:

EXAMPLE **2.3.1** Let the input matrix be the $7 \times 7$ block matrix

$$A = \begin{pmatrix} I & 0 & & & & & \\ 0 & D_1 & F_1 & & & & \\ & E_2 & D_2 & F_2 & & & \\ & & E_3 & D_3 & 0 & & \\ & & & 0 & I & 0 & \\ & & & & 0 & I & 0 \\ & & & & & 0 & I \end{pmatrix},$$

with $2 \times 2$ blocks defined by:

$$D_1 = \begin{pmatrix} 5 & 0 \\ 0 & 10 \end{pmatrix}, \quad D_2 = \begin{pmatrix} 6 & 0 \\ 0 & 5 \end{pmatrix}, \quad D_3 = \begin{pmatrix} 4 & 0 \\ 0 & 6 \end{pmatrix},$$

$$F_1 = F_2 = E_2 = \begin{pmatrix} 0 & 0 \\ 0 & 30 \end{pmatrix}, \quad E_3 = \begin{pmatrix} 0 & 0 \\ 0 & 17 \end{pmatrix}.$$

This matrix is strongly nonsingular, yet, the first diagonal block, $D_1^{(1)} = \begin{pmatrix} 6 & 0 \\ 0 & 0 \end{pmatrix}$, of the matrix $A^{(1)}$ (the coefficient matrix of the linear system obtained after the first reduction stage) is singular.

## 2.4 Computing |det $A$| via block cyclic reduction.

We extend the block cyclic reduction algorithm to computing |det $A$|, for any banded matrix $A$. We will first compute det $A$ for an h.p.d. block tridiagonal matrix $A$; then we will show how to apply the BCR algorithm to the computation of |det $A$|, for any matrix $A$ (including singular matrices).

THEOREM **2.4.1** *The cost of computing* det $A$, *for any h.p.d.* $q \times q$ *block tridiagonal matrix with* $w \times w$ *blocks, is bounded by*

$$O\left((\log q)\left(t_{D(w)} + t_{I(w)}\right), \frac{q}{\log q}\left(p_{D(w)} + p_{I(w)}\right)\right), \tag{2.20}$$

*which turns into*

$$O\left(\left(\log \frac{n}{w}\right)\left(t_{D(w)} + t_{I(w)}\right), \frac{\frac{n}{w}}{\log \frac{n}{w}}\left(p_{D(w)} + p_{I(w)}\right)\right), \tag{2.21}$$

*for* $q = n/w$.

PROOF. As in the previous sections, we assume that $q = 2^r - 1$; let $A = A^{(r)}, A^{(r-1)}, \ldots, A^{(1)}$ denote the input matrix and the coefficient matrices of the auxiliary linear systems $S^{(r)}, S^{(r-1)}, \ldots, S^{(1)}$ computed by the block cyclic reduction algorithm whereas $E_i^{(h)}$, $D_j^{(h)}$, $F_k^{(h)}$ $(1 \leq h \leq r)$ denote the blocks of $A^{(h)}$, for an appropriate choice of the indices $i$, $j$ and $k$. We first establish a recurrence relation between $\det A^{(r)}$ and $\det A^{(r-1)}$; towards this goal, we perform the following computations:

ALGORITHM **2.4.1 (BCR-determinant-reduce).**

**Input:** *An h.p.d.* $q \times q$ *block tridiagonal matrix* $A = A^{(r)}$ *with* $w \times w$ *blocks, of the format (2.5), where* $q = 2^r - 1$.

**Output:** *A* $q' \times q'$ *block tridiagonal matrix* $A^{(r-1)}$ *with* $w \times w$ *blocks, where* $q' = \frac{q-1}{2} = 2^{r-1} - 1$, *satisfying the relation*

$$\det A = \det A^{(r)} = \left( \prod_{i=1}^{q''} \det D_{2i-1}^{(r)} \right) \det A^{(r-1)}, \tag{2.22}$$

*where* $q'' = \frac{q-1}{2} + 1$.

**Computations:**

**for** $j = 1$ **to** $q'$ **do** (concurrently, for all values of $j$)

1. *premultiply the block row* $2j - 1$ *of A by the matrix* $(-E_{2j} D_{2j-1}^{-1})$ *to form a new block row denoted* $\rho_1(j)$;

2. *premultiply the block* $2j + 1$ *of A by the matrix* $(-F_{2j} D_{2j+1}^{-1})$ *to form a new block row denoted* $\rho_2(j)$;

3. *alter the block row* $2j$ *of* $A^{(r)} = A$ *to form a new matrix denoted* $\tilde{A} = \tilde{A}^{(r)}$ *by adding to it the block rows* $\rho_1(j)$ *and* $\rho_2(j)$ *[note that* $\det \tilde{A}^{(r)} = \det A^{(r)}$*];*

4. *output the matrix* $A^{(r-1)}$ *[the submatrix of the matrix* $\tilde{A}^{(r)}$ *formed by its even rows and its even columns, the first row/column being odd].* ∎

In order to prove that the relation (2.22) holds, observe that $\det \tilde{A}^{(r)} = \det A^{(r)}$) (see Proposition 1.2.2, page 8). The evaluation of $\det \tilde{A}^{(r)}$ via Definition 1.2.6 (see page 8) immediately implies (2.22). Note that the matrix $A^{(r-1)}$ is the same in both Algorithms 2.1.1 and 2.4.1. As discussed earlier, this matrix is h.p.d.; consequently, the formula (2.22) can be recursively expanded, leading to the following BCR algorithm for computing the determinant of any h.p.d. matrix:

ALGORITHM **2.4.2 (BCR-determinant).**

**Input:** *an h.p.d. q-w-block-3DM of the format (2.5), denoted* $A = A^{(r)}$.

**Output:** $\det A = \det A^{(r)}$.

**Computations:**

1. *compute the set of* $w \times w$ *matrices defined by relations*

$$\begin{aligned} \Gamma'_{bcr}(A^{(r)}) &= \begin{cases} \left\{A^{(r)}\right\}, & r = 1; \\ \gamma'_{bcr}(A^r) \bigcup \Gamma'_{bcr}\left(A^{(r-1)}\right), & r > 1; \end{cases} \\ \gamma'_{bcr}(A^{(r)}) &= \left\{D_1^{(r)}, D_3^{(r)}, \ldots, D_q^{(r)}\right\}; \end{aligned} \tag{2.23}$$

*Note that the set* $\Gamma'_{bcr}(A)$ *includes all the odd-indexed diagonal blocks of the matrices* $A^{(k)}$ $(1 \leq k \leq r)$ *[compare with the set* $\Gamma_{bcr}(A)$ *of (2.17)].*

2. *compute and output* $\det A = \prod_{X_i \in \Gamma'(A)} \det X_i$. ∎

The correctness of this algorithm follows from the equation (2.22) and its recursive expansion. In order to estimate its computational cost, observe that this evaluation of $\det A$ amounts to first computing all the auxiliary matrices $A^{(h)}$ $(1 \leq h \leq r-1)$, so that the overall cost of computing the set $\Gamma'(A)$ (stage 1 of of Algorithm 2.4.2) is bounded by

$$O\left((\log q)t_{I(w)}, \frac{q}{\log q}p_{I(w)}\right). \tag{2.24}$$

Estimating the cost of performing stage 2 of Algorithm 2.4.2, assume that we may store the set $\Gamma'(A)$ in a one-dimensional array, say, $X = [X_1, \ldots, X_q]$; then, we can compute the product

$$\det A = \prod_{Z_i \in \Gamma'(A)} \det Z_i = \prod_{i=1}^{q} \det X_i$$

at a cost bounded by

$$O\left((\log q)t_{D(w)}, \frac{q}{\log q}p_{D(w)}\right), \tag{2.25}$$

by applying the parallel prefix algorithm (see [KR], [BP] or Proposition 1.4.1, page 20). The bound (2.20) on the overall cost of computing $\det A$ follows by combining the two latter bounds. ∎

REMARK **2.4.1** In order to compute $|\det A|$ for any banded matrix $A$, we may utilize the symmetrization technique, discussed earlier in §2.3, by shifting from $A$ to $U = A^H A$. This is a valid transition, due to Propositions 1.3.1 (page 10), and the equation $\det(A^H A) = |\det{}^2 A|$. If the matrix $A$ is nonsingular, then the matrix $U$ is h.p.d.; therefore, our algorithm for computing $\det A$ in the h.p.d. case can be applied to the computation of $\det U$. This algorithm can be slightly modified to also compute $\det U$ in the case where $U$ (and therefore, also $A$) is singular: toward this end, we concurrently compute, prior to the computation of the matrix $U^{(r-1)}$ (see stage 1 of Algorithm 2.4.2), the determinants of all the odd-indexed diagonal blocks of the matrix $U^{(r)}$. If all these determinants are nonzero, then we repeat the same computations with the matrices $U^{(r-1)}$, $U^{(r-2)}$,..., and so on. Note that Algorithm 2.4.1 is applicable, as long as all these determinants are nonzero. If one of these determinants turns out to be zero, we output $\det A = 0$ and stop. Otherwise, if all the odd-indexed diagonal blocks of all the matrices $U^{(k)}$ $(1 \leq k \leq r)$ are nonsingular, then $\det U \neq 0$, and our algorithm correctly computes $|\det A|^2 = \det U$. These results follow, due to (2.22) and its recursive expansion.

The modifications of the BCR algorithm for the purpose of detecting singularity of its input matrix requires some extra computations at each parallel step. However, the cost of these additional computations does not increase the overall asymptotic bound (2.20) on the cost of computing $\det A$,

since, in each parallel step, the cost of these additional computations does not increase the asymptotic cost of performing Algorithm 2.4.1, which is required at each parallel step. Consequently, the bound (2.20) remains a valid bound on the overall cost of computing |det $A$|, for any $q \times q$ matrix $A$ that has the bandwidth $w$. ∎

# Chapter 3

## Parallel solution for a BLS over any field, based on a $3 \times 3$ block matrix representation

The block cyclic reduction led us in chapter 2 to $NC$ and optimal algorithms for the problems $BAND \cdot LIN \cdot SOLVE$ and $BAND \cdot DET$ over the fields of characteristic zero. Can we extend these algorithms to the case of finite fields ? The answer to this question does not appear obvious. In this chapter we explore a different approach to solving these problems, based on a $3 \times 3$ representation and factorization of a banded matrix. This alternative approach was utilized in a recent paper by W. Eberly [E], to derive the record bounds on the computational complexity of these two problems over an arbitrary field of constants: assuming an $n \times n$ nonsingular input matrix $M$ having bandwidth $w$, the algorithms of [E] reach the Las Vegas randomized bounds

$$\left(t_{L \cdot S(n,w)}\ ,\ p_{L \cdot S(n,w)}\right) = O\left(T^*(n), P^*(n,w)\right), \tag{3.1}$$

for the problem $BAND \cdot LIN \cdot SOLVE$ and

$$\left(t_{D(n,w)}\ ,\ p_{D(n,w)}\right) = O\left(T^*(n), P^*(n,w)\right), \tag{3.2}$$

for the problem $BAND \cdot DET$, where

$$\begin{aligned} T^*(n) &= (\log^3 n)\psi(\mathbf{F}, n)t_{I(n)}; \\ P^*(n,w) &= \left(\frac{n}{w}\right) p_{I(w)} \log^{O(1)} n; \\ \psi(\mathbf{F}, n) &= \begin{cases} 1, & \text{if } |\mathbf{F}| \geq n; \\ \left(1 + \log\log_{|\mathbf{F}|} n\right) \log_{|\mathbf{F}|} n, & \text{otherwise.} \end{cases} \end{aligned}$$

We will show a substantial improvement of these estimates of [E] to the Las Vegas randomized bounds

$$O\left(\left(\log \frac{n}{w}\right) t_{I(w)},\ \left(\frac{n}{w}\right) p_{I(w)}\right), \tag{3.3}$$

for the problem $BAND \cdot LIN \cdot SOLVE$ and

$$O\left(\left(\log \frac{n}{w}\right)\left(t_{I(w)} + t_{D(w)}\right),\ \frac{\left(\frac{n}{w}\right)\left(p_{I(w)} + p_{D(w)}\right)}{\left(\log \frac{n}{w}\right)}\right), \tag{3.4}$$

for the problem $BAND \cdot DET$.

The algorithm for $BAND \cdot LIN \cdot SOLVE$ can be extended to accelerate the computation of the solution of several linear systems, each having the same coefficient matrix $M$, by utilizing an approach similar to one of our chapter 2, that is, we first preprocess the matrix $M$ at a cost bounded by (3.3); after that, we solve any banded linear system having coefficient matrix $M$, at a *deterministic* cost bounded by

$$O\left(\left(\log \frac{n}{w}\right)(\log w),\ \frac{wn}{\log w}\right), \tag{3.5}$$

and solve $q$ linear systems, each having the same coefficient matrix $M$, at a deterministic cost bounded by

$$O\left(\left(\log \frac{n}{w}\right)(\log w), \frac{qwn}{\log w}\right). \tag{3.6}$$

The bound (3.3) [respectively, (3.4)] implies that the solution to the problem $BAND \cdot LIN \cdot SOLVE$ [respectively, $BAND \cdot DET$] is *efficient* [respectively, *optimal*] according to the definitions of [KR]. The *potential work* bounds,

$$\mathrm{w}_{L \cdot S}(n, w) = O\left(\left(\log \frac{n}{w}\right) \frac{n}{w} t_{I(w)} p_{I(w)}\right),$$

in the case of $BAND \cdot LIN \cdot SOLVE$, and

$$\mathrm{w}_{DET}(n, w) = O\left(\frac{n}{w}\left(t_{I(w)} + t_{D(w)}\right)\left(p_{I(w)} + p_{D(w)}\right)\right),$$

in the case of $BAND \cdot DET$, are better than the ones of the algorithm of [E], by the factors $\log^2 n$ for $BAND \cdot LIN \cdot SOLVE$ and $\log^3 n$ for $BAND \cdot DET$.

If the input matrix is over a field of characteristic zero, then the bounds (3.3) and (3.6) can be obtained deterministically; moreover, the problem $|BAND \cdot DET|$ (the problem of computing $|\det M|$) can be solved deterministically at a cost bounded by (3.4).

To obtain these improvements, we have combined the techniques of [E] with several other new techniques. Following the general approach, outlined in §1.7, we will divide the main algorithm (the one for the problem $BAND \cdot LIN \cdot SOLVE$) into a preprocessing stage and a backsolving stage. At the preprocessing stage, we compute a set of parameters denoted $\Gamma_3(M)$

(based on a $3 \times 3$ block representation), implicitly defining $M^{-1}$. At the backsolving stage, we utilize the set $\Gamma_3(M)$ to effectively compute the solution vector $M^{-1}\overrightarrow{\mathbf{b}}$. Our set $\Gamma_3(M)$ consists of the first $w(M)$ and the last $w(M)$ columns of the matrices $M^{-1}_{i_1 i_1 \dots i_k i_k}$, where $M_{i_1 i_1 \dots i_k i_k}$ $(k \geq 0)$ denote selected principal submatrices of $M$ [see (3.15)]. The main algorithm requires nonsingularity of the matrices $M_{i_1 i_1 \dots i_k i_k}$, which we ensure, by using the symmetrization or randomization techniques of [Sc] and [Zi], at a substantially lower computational cost than that would have been required by the techniques of [E], since the former techniques enable us to avoid the auxiliary computations of the permutation matrices of the algorithm of [E] (see §3.1).

We organize our presentation in the following order: After a brief review of the algorithm of [E] in §3.1, we present some preliminary results on the computation of the (first and/or last columns) of the the matrices $M^{-1}_{i_1 i_1 \dots i_k i_k}$ in §3.2. Then, in §3.3, we preprocess a strongly nonsingular matrix. We utilize the output of this preprocessing in §3.4 to effectively compute the solution to the input linear system(s). In §3.5, we relax the assumption about strong nonsingularity of the input matrix and, in §3.6, we compute the determinant of a banded matrix.

## 3.1 Eberly's algorithm.

Hereafter, $M$ denotes an $n \times n$ nonsingular banded matrix having bandwidth $w$, defined over an arbitrary field of constants and $\overrightarrow{\mathbf{b}}$ denotes an

$n$-dimensional vector. We briefly recall a recent algorithm by W. Eberly [E], applied to solving the linear system $M\vec{x} = \vec{b}$. Following [E], we will assume (with no loss of generality) that $n$ divides $w$, will set $\hat{n} = n/w$ and will use the following $3 \times 3$ block matrix representation, hereafter referred to as *representation-E*:

$$M = \begin{array}{c} \begin{pmatrix} M_{11} & M_{12} & 0 \\ M_{21} & M_{22} & M_{23} \\ 0 & M_{32} & M_{33} \end{pmatrix} \\ \begin{matrix} n_1 & 2w & n_3 \end{matrix} \end{array} \begin{array}{c} n_1 \\ 2w \\ n_3 \end{array} \tag{3.7}$$

where

$$n_1 = \lceil (\hat{n} - 2)/2 \rceil w, \quad n_2 = \lfloor (\hat{n} - 2)/2 \rfloor w, \tag{3.8}$$

so that $n = n_1 + n_2 + 2w$ [compare with the representation (3.13)].

In order to compute the vector $M^{-1}\vec{b}$, the algorithm of [E] first computes a recursive decomposition of the matrix $M^{-1}$, based on representation-E [the components of this decomposition are either sparse matrices or dense matrices of size at most $(2w) \times (2w)$]; after that, the vector $M^{-1}\vec{b}$ is recovered via successive (sparse) matrix-by-vector multiplications.

The technique of [E] requires nonsingularity of all the diagonal blocks in representation-E of matrix $M$, which is ensured via permutations of some selected rows of $M$. The cost of computing the required permutations and other related results can be summarized as follows:

PROPOSITION **3.1.1** ([E]). *Let $M$ denote an $n \times n$ nonsingular matrix having bandwidth $w$.*

1. *There exists a permutation matrix of the format*

$$P_{fix} = \text{diag}\,(I_{n_1-w}, P_T, P_B, I_{n_2-w}), \tag{3.9}$$

*where $P_T$ and $P_B$ are $w \times w$ permutation matrices, such that, in representation (3.7) of the matrix*

$$P_{fix}M = \begin{pmatrix} M_1 & B_U & 0 \\ B_L & X & B_R \\ 0 & B_D & M_2 \end{pmatrix}, \tag{3.10}$$

*the diagonal blocks $M_1$, $X$ and $M_2$, which, in representation-E of the matrix $P_{fix}M$, correspond to the blocks $M_{11}$, $M_{22}$ and $M_{33}$, respectively, are nonsingular.*

2. *Moreover, the permutation matrices $P_T$ and $P_B$ can be chosen so that the bandwidth of each of the matrices $M_1$ and $M_2$ does not exceed $w$.*

3. *Furthermore, a required permutation matrix $P_{fix}$ can be computed (by means of a Las Vegas randomized algorithm) at a cost bounded by*

$$O\left(\left(\log^2 n\right) t_{I(n)}, \left(\frac{n}{w}\right) p_{I(w)}\right). \tag{3.11}$$

■

Due to nonsingularity of the diagonal blocks $M_1$, $X$, $M_2$ of (3.10), the matrices $M$ and $M^{-1}$ have factorizations of the following format, hereafter referred to as to *factorization-E*:

$$M = P_{fix}^{-1}P_L L D U P_L^{-1}, \qquad M^{-1} = P_L U^{-1} D^{-1} L^{-1} P_L^{-1} P_{fix}, \tag{3.12}$$

$$L = \begin{pmatrix} I_{n_1} & 0 & 0 \\ 0 & I_{n_2} & 0 \\ B_L M_1^{-1} & B_R M_2^{-1} & I_{2w} \end{pmatrix}, \qquad L^{-1} = \begin{pmatrix} I_{n_1} & 0 & 0 \\ 0 & I_{n_2} & 0 \\ -B_L M_1^{-1} & -B_R M_2^{-1} & I_{2w} \end{pmatrix},$$

$$U = \begin{pmatrix} I_{n_1} & 0 & M_1^{-1} B_U \\ 0 & I_{n_2} & M_2^{-1} B_D \\ 0 & 0 & I_{2w} \end{pmatrix}, \qquad U^{-1} = \begin{pmatrix} I_{n_1} & 0 & -M_1^{-1} B_U \\ 0 & I_{n_2} & -M_2^{-1} B_D \\ 0 & 0 & I_{2w} \end{pmatrix},$$

$$D = \operatorname{diag}(M_1, M_2, Y), \qquad D^{-1} = \operatorname{diag}(M_1^{-1}, M_2^{-1}, Y^{-1}),$$

$$P_L = \begin{pmatrix} I_{n_1} & 0 & 0 \\ 0 & 0 & I_{2w} \\ 0 & I_{n_2} & 0 \end{pmatrix}, \qquad P_L^{-1} = \begin{pmatrix} I_{n_1} & 0 & 0 \\ 0 & 0 & I_{n_2} \\ 0 & I_{2w} & 0 \end{pmatrix},$$

$$Y = X - B_L M_1^{-1} B_U - B_R M_2^{-1} B_D.$$

Since the matrices $M_1$ and $M_2$ can be decomposed the same way, this decomposition can be continued recursively.

PROPOSITION **3.1.2** ([E]). *Any nonsingular banded matrix $M$ having bandwidth $w$, as well as its inverse $M^{-1}$, can be expressed as a product of $O\left(\log \frac{n}{w}\right)$ matrices, each of which is either a permutation matrix, a sparse lower triangular matrix (of the format L), a sparse upper triangular matrix (of the format U), a dense nonsingular matrix of size $2w \times 2w$, or is block diagonal with each diagonal having these forms.*

*Moreover, such decompositions of $M$ and $M^{-1}$ can be computed by a Las Vegas randomized parallel algorithm, at a cost bounded by (3.1).*

Once the decompositions $M$ and $M^{-1}$ of the format described in Proposition 3.1.2 have been computed, the vector $M^{-1}\overrightarrow{\mathbf{b}}$ and the determinant $\det M$ can be effectively computed at a cost bounded by (3.1) [the actual cost of the subsequent computation of the vector $M^{-1}\overrightarrow{\mathbf{b}}$ is lower than (3.1) [see §3.4 for a similar computation].

REMARK **3.1.1 (Is it mandatory to compute $P_{fix}$ ?).** The sole purpose of computing the permutation $P_{fix}$ is to ensure nonsingularity of the diagonal blocks in representation (3.7) of matrix M, so as to derive factorization-3 for matrix $M$. However the cost (3.11) of computing this permutation matrix is relatively high. Indeed, in §3.5, we present alternative techniques for ensuring nonsingularity of these diagonal blocks at a lower computational cost.

REMARK **3.1.2 (Block cyclic reduction versus Eberly's algorithm).** According to the definitions of [KR], the block cyclic reduction algorithm is optimal (assuming the *CREW PRAM* model of parallel computations), but it is restricted to the fields of characteristic zero. On the other hand, the algorithm of [E] is efficient (but not optimal); however, it is more general, since it is applicable over any field of constants.

## 3.2 Some definitions and preliminary results.

We define *representation-3* of matrix $M$ as the $3 \times 3$ block matrix representation

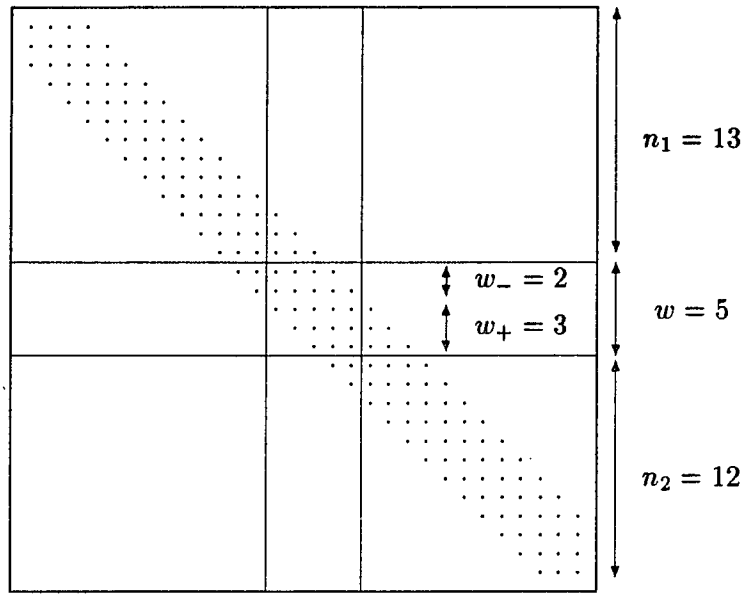



Figure 3.1: Representation-3 of a $30 \times 30$ banded matrix.

$$M = \begin{pmatrix} M_{11} & M_{12} & 0 \\ M_{21} & M_{22} & M_{23} \\ 0 & M_{32} & M_{33} \end{pmatrix} \begin{matrix} n_1 \\ w \\ n_3 \end{matrix} \qquad \begin{matrix} \\ \\ \\ n_1 \quad w \quad n_3 \end{matrix} \tag{3.13}$$

where

$$n_1 = \lceil (n-w)/2 \rceil, \quad n_3 = n - w - n_1 \tag{3.14}$$

[see Figure 3.1, compare with the representation of [E], (3.7)].

$M_{i_1 i_1 \ldots i_k i_k}$ $(k \geq 0)$ denotes the principal submatrices of $M$ defined by the relation

$$M_{i_1 i_1 \ldots i_k i_k} = \begin{cases} M & \text{if } k = 0, \\ (M_{i_1 i_1 \ldots i_{k-1} i_{k-1}})_{i_k i_k} & \text{otherwise,} \end{cases} \tag{3.15}$$

where $i_1, \ldots, i_k \in \{1,2,3\}$ and $(M_{i_1 i_1 \ldots i_{k-1} i_{k-1}})_{i_k i_k}$ denotes the $(i_k, i_k)$ block in the representation-3 of matrix $M_{i_1 i_1 \ldots i_{k-1} i_{k-1}}$.

Until §3.5, we will assume that the matrix $M$ is *strongly nonsingular*, so that $M$, $M^{-1}$ and all their principal submatrices have factorizations of the following format, hereafter referred to as to *factorization-3* of M:

$$M = V \begin{pmatrix} M_{11} & 0 & 0 \\ 0 & Z & 0 \\ 0 & 0 & M_{33} \end{pmatrix} W, \tag{3.16}$$

$$M^{-1} = W^{-1} \begin{pmatrix} M_{11}^{-1} & 0 & 0 \\ 0 & Z^{-1} & 0 \\ 0 & 0 & M_{33}^{-1} \end{pmatrix} V^{-1}, \tag{3.17}$$

where

$$V = \begin{pmatrix} I_{n_1} & 0 & 0 \\ M_{21}M_{11}^{-1} & I_m & M_{23}M_{33}^{-1} \\ 0 & 0 & I_{n_3} \end{pmatrix}, \quad W = \begin{pmatrix} I_{n_1} & M_{11}^{-1}M_{12} & 0 \\ 0 & I_m & 0 \\ 0 & M_{33}^{-1}M_{32} & I_{n_3} \end{pmatrix},$$

$$V^{-1} = \begin{pmatrix} I_{n_1} & 0 & 0 \\ -M_{21}M_{11}^{-1} & I_m & -M_{23}M_{33}^{-1} \\ 0 & 0 & I_{n_3} \end{pmatrix}, \quad W^{-1} = \begin{pmatrix} I_{n_1} & -M_{11}^{-1}M_{12} & 0 \\ 0 & I_m & 0 \\ 0 & -M_{33}^{-1}M_{32} & I_{n_3} \end{pmatrix},$$

$$Z = M_{22} - M_{21}M_{11}^{-1}M_{12} - M_{23}M_{33}^{-1}M_{32}. \tag{3.18}$$

Note that $\det V = \det W = 1$; therefore, the equation (3.16) implies that

$$\det M = (\det M_{11})\,(\det Z)\,(\det M_{33}), \tag{3.19}$$

which implies nonsingularity of matrix $Z$.

The preprocessing algorithm will need to compute the submatrices of the matrices $M^{-1}_{i_1 i_1 \ldots i_k i_k}$ formed by their first $w$ and their last $w$ columns. We will compute these submatrices by using two auxiliary matrices denoted

$$I_F(k,p) = \begin{pmatrix} I_p & 0 \\ 0 & 0 \end{pmatrix}, \tag{3.20}$$

$$I_L(k,p) = \begin{pmatrix} 0 & 0 \\ 0 & I_p \end{pmatrix}. \tag{3.21}$$

where $k > p$ and where we set $p = w$.

PROPOSITION **3.2.1** *The matrices $I_F(k,p)$ and $I_L(k,p)$ satisfy the following properties, where $r$, $s$, $r_1$, $s_1$, $r_2$, and $s_2$ are positive integers such that $r = r_1 + r_2$, $s = s_1 + s_2$, $p \leq r$, $p \leq s$, $B$ is a $p \times p$ matrix, $W_{11}$ is an $r_1 \times s_1$ matrix and $W = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix}$ is an $r \times s$ matrix:*

$$I_F(r,r_1)W = \begin{pmatrix} W_{11} & W_{12} \\ 0 & 0 \end{pmatrix}, \tag{3.22}$$

$$I_L(r,r_2)W = \begin{pmatrix} 0 & 0 \\ W_{21} & W_{22} \end{pmatrix}, \tag{3.23}$$

$$WI_F(s,s_1) = \begin{pmatrix} W_{11} & 0 \\ W_{21} & 0 \end{pmatrix}, \tag{3.24}$$

$$WI_L(s,s_2) = \begin{pmatrix} 0 & W_{12} \\ 0 & W_{22} \end{pmatrix}, \tag{3.25}$$

$$I_F(r,r_1)WI_F(s,s_1) = \begin{pmatrix} W_{11} & 0 \\ 0 & 0 \end{pmatrix}, \tag{3.26}$$

$$I_F(r,r_1)WI_L(s,s_2) = \begin{pmatrix} 0 & W_{12} \\ 0 & 0 \end{pmatrix}, \tag{3.27}$$

$$I_L(r,r_2)WI_F(s,s_1) = \begin{pmatrix} 0 & 0 \\ W_{21} & 0 \end{pmatrix}, \tag{3.28}$$

$$I_L(r,r_2)WI_L(s,s_2) = \begin{pmatrix} 0 & 0 \\ 0 & W_{22} \end{pmatrix}; \tag{3.29}$$

*moreover, if* $(B \quad 0)$ *and* $(0 \quad B)$ *are* $p \times r$ *matrices and* $\binom{B}{0}$, $\binom{0}{B}$ *are* $s \times p$ *matrices, then*

$$(B \quad 0) = (B \quad 0)\, I_F(r,p), \tag{3.30}$$

$$(0 \quad B) = (0 \quad B)\, I_L(r,p), \tag{3.31}$$

$$\binom{B}{0} = I_F(s,p)\binom{B}{0}, \tag{3.32}$$

$$\binom{0}{B} = I_L(s,p)\binom{0}{B}. \tag{3.33}$$

The above relations are immediately verified by simple inspection.

REMARK **3.2.1** For any $k \times k$ matrix $W$ and any $p \leq k$, we may use the matrix $WI_F(k,p)$ [respectively, $WI_L(k,p)$] to represent the $k \times p$ submatrix of $W$ formed by its first (respectively, last) $p$ columns, since, due to (3.24) and (3.25), the entries of the matrix $WI_F(k,p)$ (respectively, $WI_L(k,p)$) are filled with zeros, except for its first (respectively, last) $p$ columns, which are filled with the corresponding entries of $W$. Likewise, we may use the matrix $I_F(k,p)W$ [respectively, $I_L(k,p)W$] to represent the submatrix of $W$ formed by its first (respectively, last) $p$ rows, since, due to (3.22) and (3.23), the entries of the matrix $I_F(k,p)W$ (respectively, $I_L(k,p)W$) are filled with zeros, except for its first (respectively, last) $p$ rows, which are filled with the corresponding entries of $W$. The indices $F$ and $L$ stand for "First" and "Last", respectively.

PROPOSITION **3.2.2** *It follows from Proposition 3.2.1 that the blocks (1,2), (2,1), (2,3) and (3,2) of M in representation-3 satisfy the equations*

$$M_{12} = I_L(n_1, w)M_{12}, \tag{3.34}$$

$$M_{21} = M_{21}I_L(n_1, w), \tag{3.35}$$

$$M_{23} = M_{23}I_F(n_3, w), \tag{3.36}$$

$$M_{32} = I_F(n_3, w)M_{32}. \tag{3.37}$$

PROPOSITION **3.2.3** *Once the first w and the last w columns of the matrices $M_{11}^{-1}$ and $M_{22}^{-1}$ have been computed, the subsequent computation of the matrices Z and $Z^{-1}$ can be performed at costs bounded by*

$$O\left(t_{M(w,w,w)},\ p_{M(w,w,w)}\right) \tag{3.38}$$

*and*

$$O\left(t_{I(w)},\ p_{I(w)}\right), \tag{3.39}$$

*respectively.*

PROOF. Due to Proposition 3.2.2, we may substitute in equation (3.18) $I_L(n_1, w)M_{12}$, $M_{21}I_L(n_1, w)$, $M_{23}I_F(n_3, w)$ and $I_F(n_3, w)M_{32}$] for $M_{12}$, $M_{21}$, $M_{23}$ and $M_{32}$, respectively, to derive the equation

$$\begin{aligned} Z = M_{22} &- M_{21}I_L(n_1, w)M_{11}^{-1}I_L(n_1, w)M_{12} \\ &- M_{23}I_F(n_3, w)M_{33}^{-1}I_F(n_3, w)M_{32}. \end{aligned} \tag{3.40}$$

Since the first $w$ and the last $w$ columns of the matrices $M_{11}^{-1}$ and $M_{33}^{-1}$ are assumed computed, the matrices $M_{11}^{-1}I_F(n_1, w)$, $M_{11}^{-1}I_L(n_1, w)$, $M_{33}^{-1}I_F(n_3, w)$

and $M_{33}^{-1} I_L(n_3, w)$ are readily available, and, due to the equations (3.40) and (3.26)–(3.29), the computation of the matrix $Z$ only requires four $w \times w$ matrix multiplications. The bound (3.38) on the cost of computing the matrix $Z$ and the bound (3.39) on the cost of computing $Z^{-1}$ immediately follow. ∎

PROPOSITION **3.2.4** *Once the first $w$ and the last $w$ columns of the matrices $M_{11}^{-1}$ and $M_{22}^{-1}$ have been computed, the complexity of the subsequent computation of the first $w$ and the last $w$ columns of the matrix $M^{-1}$ is bounded by*

$$O\left(t_{I(w)}, \left(\frac{n}{w}\right) p_{I(w)}\right). \tag{3.41}$$

PROOF. Expand (3.17), then, postmultiply both sides of the resulting equation by $I_F(n, w)$ and $I_L(n, w)$, respectively, and apply Proposition 3.2.2 to derive the following equations

$$\begin{aligned} M^{-1} I_F(n, w) &= \begin{pmatrix} M_{11}^{-1} I_F(n_1, w) + M_{11}^{-1} I_L(n_1, w) M_{12} Z^{-1} M_{21} M_{11}^{-1} I_F(n_1, w) & 0 \\ -Z^{-1} M_{21} M_{11}^{-1} I_F(n_1, w) & 0 \\ M_{33}^{-1} I_F(n_3, w) M_{32} Z^{-1} M_{21} M_{11}^{-1} I_F(n_1, w) & 0 \end{pmatrix} \\ &= \begin{pmatrix} F_1' + L_1' M_{12} Z^{-1} M_{21} F_1' & 0 \\ -Z^{-1} M_{21} F_1' & 0 \\ F_3' M_{32} Z^{-1} M_{21} F_1' & 0 \end{pmatrix} \end{aligned} \tag{3.42}$$

and

$$M^{-1}I_L(n,w) = \begin{pmatrix} 0 & M_{11}^{-1}I_L(n_1,w)M_{12}Z^{-1}M_{23}M_{33}^{-1}I_L(n_3,w) \\ 0 & -Z^{-1}M_{23}M_{33}^{-1}I_L(n_3,w) \\ 0 & M_{33}^{-1}I_L(n_3,w) + M_{33}^{-1}I_F(n_3,w)M_{32}Z^{-1}M_{23}M_{33}^{-1}I_L(n_3,w) \end{pmatrix}$$

$$= \begin{pmatrix} 0 & L_1'M_{12}Z^{-1}M_{23}L_3' \\ 0 & -Z^{-1}M_{23}L_3' \\ 0 & L_3' + F_3'M_{32}Z^{-1}M_{23}L_3' \end{pmatrix}, \tag{3.43}$$

respectively, where

$$F_1' = M_{11}^{-1}I_F(n_1,w), \tag{3.44}$$

$$L_1' = M_{11}^{-1}I_L(n_1,w), \tag{3.45}$$

$$F_3' = M_{33}^{-1}I_F(n_3,w), \tag{3.46}$$

$$L_3' = M_{33}^{-1}I_L(n_3,w). \tag{3.47}$$

Since the first $w$ and the last $w$ columns of the matrices $M_{11}^{-1}$ and $M_{33}^{-1}$ are assumed computed, the matrices $F_1'$, $L_1'$, $F_3'$ and $L_3'$ are readily available; moreover, due to Proposition 3.2.3, the matrix $Z^{-1}$ can be computed at a cost bounded by $O\left(t_{I(w)}, p_{I(w)}\right)$. After that, due to the equations (3.42) and (3.43), the computation of $M^{-1}I_F(n,w)$, $M^{-1}I_L(n,w)$ only requires a constant number of $q \times w$ by $w \times w$ matrix multiplications, where $q \leq \lceil n/2 \rceil$. The bound (3.41) on the computation of the first $w$ and the last $w$ columns of the matrix $M^{-1}$ follows. ∎

## 3.3 Preprocessing a strongly nonsingular BLS, based on a $3 \times 3$ block representation of the input matrix.

Hereafter, $Z_{i_1 i_1 \ldots i_k i_k}$ denotes the matrix corresponding to $Z$ in factorization-3 of matrix $M_{i_1 i_1 \ldots i_k i_k}$, for $i_1, \ldots, i_k \in \{1, 3\}$; $\Gamma_3(M)$ denotes the set of parameters computed by the preprocessing algorithm and is defined by the relations

$$\Gamma_3(M) = \begin{cases} \{M^{-1}\}, & n < 2w, \\ \gamma_3(M) \bigcup \Gamma_3(M_{11}) \bigcup \Gamma_3(M_{33}), & \text{otherwise}, \end{cases} \tag{3.48}$$

where

$$\gamma_3(M) = \{Z^{-1},\ M^{-1} I_F(n, w),\ M^{-1} I_L(n, w)\}; \tag{3.49}$$

note that set $\Gamma_3(M)$ consists of all the matrices $Z^{-1}_{i_1 i_1 \ldots i_k i_k}$ and all the submatrices of $M^{-1}_{i_1 i_1 \ldots i_k i_k}$ formed by their first $w$ or their last $w$ columns, for all $k \geq 0$.

Based on representation-3, we define the problem *PREPROCESS*-3, for a strongly nonsingular matrix $M$ as the computation of the set $\Gamma_3(M)$.

THEOREM **3.3.1** *The problem PREPROCESS-3, for any $n \times n$ strongly nonsingular matrix having bandwidth $w$, can be solved at a cost bounded by (3.3).*

PROOF. Due to (3.48), the set $\Gamma_3(M)$ can be evaluated by performing the following computations:

ALGORITHM **3.3.1 (Preprocessing-3).**

1. *concurrently compute the sets* $\Gamma_3(M_{11})$ *and* $\Gamma_3(M_{33})$, *by recursively applying this algorithm, unless* $n \leq 2w$, *in which case, simply output* $\Gamma_3(M) = \{M^{-1}\}$ [note that after this step, the matrices $F_1'$, $L_1'$, $F_3'$ and $L_3'$ of (3.44)–(3.47) are readily available];

2. *compute the set* $\gamma_3(M)$, *at a cost bounded by* $O\left(t_{I(w)}, \left(\frac{n}{w}\right) p_{I(w)}\right)$, *by applying Proposition 3.2.3 (see page 66) and Proposition 3.2.4 (see page 67).*

Let $t_1 = t_1(n, w)$ bound the number of parallel steps required for the overall computation of the set $\Gamma_3(M)$ and let $p_1 = p_1(n, w)$ denote the corresponding number of required processors. The above algorithm leads to the following relations for the pair $(t_1, p_1)$, estimated within a constant factor (for simplicity, we assume that $n$ is a power of 2):

$$t_1(n,w) \leq \begin{cases} t_{I(w)}, & n < 2w; \\ t_1(n/2,w) + t_{I(w)}, & \text{otherwise;} \end{cases}$$

$$p_1(n,w) \leq \begin{cases} p_{I(w)}, & n < 2w; \\ \max\left\{2p_1(n/2,w), \left(\frac{n}{w}\right) p_{I(w)}\right\}, & \text{otherwise.} \end{cases}$$

Recursive application of the latter estimates yields the complexity bound (3.3) on the pair $(p_1, t_1)$. ∎

## 3.4 Solving a preprocessed BLS: $BACK \cdot SOLVE$-3.

THEOREM **3.4.1** *Once the set $\Gamma_3(M)$ has been computed, the computational complexity of the subsequent evaluation of $M^{-1}\overrightarrow{\mathbf{b}}$, for any given n-dimensional vector $\overrightarrow{\mathbf{b}}$, is bounded by* (3.5).

PROOF. Denote $\overrightarrow{\mathbf{b}} = \left(\overrightarrow{\mathbf{b}_1}, \overrightarrow{\mathbf{b}_2}, \overrightarrow{\mathbf{b}_3}\right)^T$ where $\overrightarrow{\mathbf{b}_1}$, $\overrightarrow{\mathbf{b}_2}$ and $\overrightarrow{\mathbf{b}_3}$ are three column vectors of dimensions $n_1$, $w$ and $n_3$, respectively. Expand (3.17) then, postmultiply both sides of the resulting equation by the vector $\overrightarrow{\mathbf{b}}$ and apply Proposition 3.2.2 to derive the following equations where the matrices $L_1'$ and $F_3'$ are defined by (3.45) and (3.46), respectively:

$$
\begin{aligned}
M^{-1}\overrightarrow{\mathbf{b}} &= \begin{pmatrix} M_{11}^{-1}\overrightarrow{\mathbf{b}_1} - M_{11}^{-1}I_L(n_1,w)M_{12}Z^{-1}(-M_{21}M_{11}^{-1}\overrightarrow{\mathbf{b}_1} + \overrightarrow{\mathbf{b}_2} - M_{23}M_{33}^{-1}\overrightarrow{\mathbf{b}_3}) \\ Z^{-1}(-M_{21}M_{11}^{-1}\overrightarrow{\mathbf{b}_1} + \overrightarrow{\mathbf{b}_2} - M_{23}M_{33}^{-1}\overrightarrow{\mathbf{b}_3}) \\ M_{33}^{-1}\overrightarrow{\mathbf{b}_3} - M_{33}^{-1}I_F(n_3,w)M_{32}Z^{-1}(-M_{21}M_{11}^{-1}\overrightarrow{\mathbf{b}_1} + \overrightarrow{\mathbf{b}_2} - M_{23}M_{33}^{-1}\overrightarrow{\mathbf{b}_3}) \end{pmatrix} \\
&= \begin{pmatrix} \overrightarrow{\mathbf{u}_1} - L_1'M_{12}Z^{-1}(-M_{21}\overrightarrow{\mathbf{u}_1} + \overrightarrow{\mathbf{b}_2} - M_{23}\overrightarrow{\mathbf{u}_3}) \\ Z^{-1}(-M_{21}\overrightarrow{\mathbf{u}_1} + \overrightarrow{\mathbf{b}_2} - M_{23}\overrightarrow{\mathbf{u}_3}) \\ \overrightarrow{\mathbf{u}_3} - F_3'M_{32}Z^{-1}(-M_{21}\overrightarrow{\mathbf{u}_1} + \overrightarrow{\mathbf{b}_2} - M_{23}\overrightarrow{\mathbf{u}_3}) \end{pmatrix}; \qquad (3.50) \\
\overrightarrow{\mathbf{u}_1} &= M_{11}^{-1}\overrightarrow{\mathbf{b}_1}; \\
\overrightarrow{\mathbf{u}_3} &= M_{33}^{-1}\overrightarrow{\mathbf{b}_3}.
\end{aligned}
$$

In order to compute the vector $M^{-1}\overrightarrow{\mathbf{b}}$, we first concurrently compute the vectors $M_{11}^{-1}\overrightarrow{\mathbf{b}_1}$ and $M_{33}^{-1}\overrightarrow{\mathbf{b}_3}$; after that, due to the equation (3.50), and since the matrices $Z^{-1}$, $L_1'$ and $F_3'$ are available (from preprocessing), the

computation of the vector $M^{-1}\overrightarrow{\mathbf{b}}$ only amounts to a constant number of multiplications of matrices of size at most $w \times \lceil n/2 \rceil$ by a vector of dimension at most $\lceil n/2 \rceil$.

Let $t_2 = t_2(n,w)$ bound the number of parallel steps required for the computation of the vector $M^{-1}\overrightarrow{\mathbf{b}}$ and let $p_2 = p_2(n,w)$ denote the corresponding number of required processors (assuming the set $\Gamma_3(M)$ available). The above algorithm leads to the following relations for the pair $(t_2, p_2)$, estimated within a constant factor:

$$t_2(n,w) \leq \begin{cases} t_{M(w,w,1)}, & n \leq 2w, \\ t_2(n/2,w) + t_{M(n/2,w,1)}, & \text{otherwise;} \end{cases}$$

$$p_2(n,w) \leq \begin{cases} p_{M(w,w,1)}, & n \leq 2w, \\ \max\left(2p_2(n/2,w),\ p_{M(n/2,w,1)}\right), & \text{otherwise.} \end{cases}$$

Recursive application of the latter relations yields the complexity bound (3.5) on the pair $(p_2, t_2)$. ∎

REMARK **3.4.1 (Storage space for $PREPROCESS$-3).** The computations in each parallel step of the solutions to the problems $PREPROCESS$-3 or $BACK \cdot SOLVE$-3 only involve the elements of the set $\Gamma_3(M)$ computed in the previous parallel step. Consequently, the results of the current parallel step can overwrite the ones of the previous step, so that the storage space for this solution to these two problems is bounded by $O(wn)$.

REMARK **3.4.2 (Solving several linear systems).** Once the set $\Gamma_3(M)$ has been computed, the solution of $q$ linear systems, each having the same

coefficient matrix $M$, can be computed at a cost bounded by (3.6), by substituting two $n \times q$ matrices $X$ and $B$ for the vectors $\vec{\mathbf{x}}$ and $\vec{\mathbf{b}}$, respectively, in the proof of Theorem 3.4.1. The overall memory (storage) space required for our algorithm is bounded by $O(wn + qn)$ in this case.

## 3.5 Relaxing the strong nonsingularity assumption.

Over the fields of characteristic zero, we may use the symmetrization technique in order to relax the assumption about strong nonsingularity of the input matrix $M$, by shifting from $M$ to $(M^H M)$. The argument here is the same as the one of chapter 2, used there to relax a similar, but h.p.d. assumption about the input matrix (see §2.3, page §46).

Alternatively, in the case of any field, we will use an argument based on randomization. We will assume (with no loss of generality) that the input matrix, hereafter denoted $M'$, is an $n \times n$ nonsingular matrix of the format

$$M' = M + \text{diag } (B, 0, C), \tag{3.51}$$

where $w(M) = w$, $w_-(M) = w_+(M) = (w/2)$ and where $B$, $C$ are $w \times w$ matrices (see figure 3.2). Denote $w' = w(M')$ (note that $w \leq w' \leq 3w$), and denote $\mathcal{M}'(n, w)$ the set of matrices $M'$ satisfying the conditions listed above. It is sufficient to show how to relax the strong nonsingularity assumption for all the matrices in $\mathcal{M}'(n, w)$, since $\mathcal{M}'(n, w)$ includes all nonsingular matrices that have the bandwidth $w$.

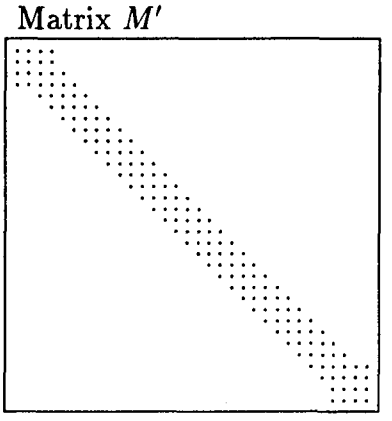



Figure 3.2: Format of the matrices $M' \in \mathcal{M}'(32,4)$.

---

PROPOSITION **3.5.1** *Let* $P = \text{diag}\left(I_{n_1-w/2}, R_1, R_3, I_{n_3-w/2}\right)$, *where* $R_1$ *and* $R_3$ *are nonsingular matrices of size* $w \times w$ *and let* $M' \in \mathcal{M}'(n,w)$. *Denote* $(PM')_{ij}$ *the* $(i,j)$ *block of* $PM'$ *in representation-3 of matrix* $PM'$. *The following properties hold:*

1. *the blocks* $(PM')_{12}$, $(PM')_{21}$, $(PM')_{23}$ *and* $(PM')_{32}$ *have the formats* $\begin{pmatrix} 0 \\ W_1 \end{pmatrix}$, $(0 \quad W_2)$, $(W_3 \quad 0)$ *and* $\begin{pmatrix} W_4 \\ 0 \end{pmatrix}$, *respectively, where the* $W_i$*'s are matrices of size* $w \times w$ *each. Consequently, Proposition 3.2.2 (see page 65) still holds if we substitute* $(PM')_{ij}$ *for* $M_{ij}$ $(i,j \in \{1,2,3\}$ *and* $i \neq j)$;

2. *if* $(PM')_{11}$ *and* $(PM')_{33}$ *are nonsingular, then* $(PM')_{11} \in \mathcal{M}'(n_1,w)$ *and* $(PM')_{33} \in \mathcal{M}'(n_3,w)$.

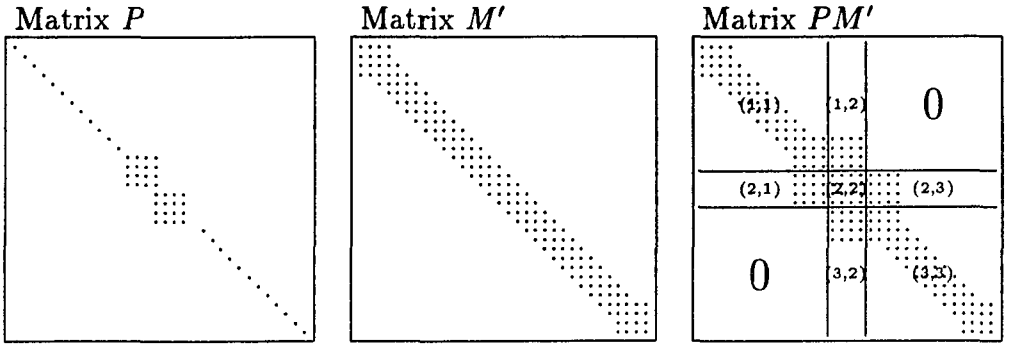



Figure 3.3: Format of the matrices $(PM')_{ij}$, where $M' \in \mathcal{M}'(32,4)$ and $P = \text{diag}\,(I_{12}, R_1, R_3 I_{14})$; here, $R_1$ and $R_3$ are $4 \times 4$ matrices.

---

This proposition follows, by a simple inspection of the product $PM'$ (see Figure 3.3). Note that $w \leq w(PM') \leq 3w$. ∎

We now show how to transform any linear system of the format

$$M'\overrightarrow{\mathbf{x}} = \overrightarrow{\mathbf{b}}, \tag{3.52}$$

where $M' \in \mathcal{M}'(n,w)$, into a suitable input to preprocessing (Algorithm 3.3.1, page 69). To this end, we will shift from (3.52) to

$$(PM')\overrightarrow{\mathbf{x}} = P\overrightarrow{\mathbf{b}}, \tag{3.53}$$

where

$$P = \text{diag}\left(I_{n_1 - w/2}, R_1, R_3, I_{n_3 - w/2}\right), \tag{3.54}$$

$R_1$, $R_3$ are random matrices of size $w \times w$. Since the matrices $R_1$ and $R_3$ are nonsingular (with a high probability), the matrix $P$ is nonsingular. Con-

sequently the equations (3.52) and (3.53) are equivalent; therefore, in order to obtain the vector $M'^{-1}\overrightarrow{\mathbf{b}}$ we may simply compute $(PM')^{-1}(P\overrightarrow{\mathbf{b}})$. The transition from (3.52) to (3.53) only amounts to a constant number of $w \times w$ matrix multiplications, which can be performed at a cost bounded by $O\left(t_{M(w,w,w)},\ p_{M(w,w,w)}\right)$. The cost of this transition increases neither the bound (3.41) on the computational complexity of each parallel step of the preprocessing algorithm, nor the asymptotic bound $O\left(t_{M(n,w,1)},\ p_{M(n,w,1)}\right)$ on the computational complexity of each parallel step of the backsolving algorithm. We claim that the auxiliary system (3.53) is a suitable input to the preprocessing algorithm. To justify this claim, observe that the algorithm actually only requires that the two following conditions be satisfied:

a) The (1,2), (2,1), (2,3) and (3,2) blocks in representation-3 of matrix $PM'$ should verify Proposition 3.2.2 (see page 65). This condition immediately follows from Proposition 3.5.1 (see page 73).

b) The matrices $PM'$, $(PM')_{11}$, $(PM')_{33}$ and $Z_{PM'}$ should be nonsingular (here, $Z_{PM'}$ denotes the matrix corresponding to $Z$ in factorization-3 of matrix $PM'$). In order to ensure this condition, we first note that $PM'$ is nonsingular (with a high probability), and since nonsingularity of $Z_{PM'}$ follows from nonsingularity of $PM'$, $(PM')_{11}$ and $(PM')_{33}$, it is sufficient to ensure nonsingularity of $(PM')_{11}$ and $(PM')_{33}$ assuming that $PM'$ is nonsingular. We show in Theorem 3.5.1 how to assign the entries of $R_1$ and $R_3$ so as to yield nonsingularity for the two larger principal diagonal blocks

$(PM')_{11}$ and $(PM')_{33}$. By the standard argument of [Sc], [Zi], nonsingularity of these two principal submatrices of $PM'$ follows (with a high probability) for a random assignment of values (from a fixed large set) to the entries of $R$.

Moreover, due to Proposition 3.5.1, the matrices $(PM')_{11}$ and $(PM')_{33}$ have the format of the original input matrix $M'$; consequently, this randomization process can be continued recursively. ■

REMARK **3.5.1** In the case of exact computations (such as rational computations or finite field computations), the preprocessing algorithm will always fail (as expected) on any singular input linear system. In the case of nonsingular input linear systems, this algorithm may occasionally fail to compute the correct answer. For instance, this will happen if one of the random matrices $R_1$ or $R_3$ turns out to be singular. Such cases, however, will only occur with a low probability, provided that the ground field has a sufficiently large cardinality.

THEOREM **3.5.1** *Let $M$ be a nonsingular banded matrix having size $n \times n$; set $w = w(M)$, $w_- = w_-(M)$, $w_+ = w_+(M)$, and let $n_1$ and $n_3$ be defined by (3.14). Then there exist two permutation matrices denoted $R_1$ and $R_3$, each of size $w \times w$ and such that, in representation-3 of matrix $Q = PM =$* $\mathrm{diag}\,(I_{n_1-w_+}, R_1, R_3, I_{n_3-w_-})M$, *the blocks $Q_{11}$ and $Q_{33}$ are nonsingular (here $Q_{ij}$ denote the blocks of $Q$ in representation-3 of matrix $Q$).*

PROOF. We will exploit an argument from [E], used in the existence proof of the permutation $P_{fix}$ (see §3.1). Let $M_{ij}$ denote the blocks in representation-3 of matrix $M$, $M_L$ denote the $n \times n_1$ matrix $\begin{pmatrix} M_{11} \\ M_{21} \\ 0 \end{pmatrix}$. Since $M$ is nonsingular, $M_L$ has full (column) rank $n_1$. The top $n_1 - w_+(M)$ rows of $M_L$ are linearly independent, since the top $n_1 - w_+(M)$ rows of $M$ are linearly independent, and since these rows have no nonzero entries outside $M_L$ (see figure 3.4). The top $n_1 + w_-(M)$ rows of $M_L$ have full rank, $n_1$, as well, because $M_L$ has this rank and has no nonzero entries below these rows. It follows that there exist $n_1$ rows of $M_L$ that include the first $n_1 - w_+(M)$ rows, as well as $w_+(M)$ of the next $w(M)$ rows, and that form a nonsingular $n_1 \times n_1$ matrix. Consequently, there exists a permutation matrix $R_1$ of size $w(M) \times w(M)$ such that the leading principal $n_1 \times n_1$ submatrix of the matrix $Q = \text{diag}\,(I_{n_1 - w_+}, R_1, I_{n_3 + w_-})M$ is nonsingular. A similar argument shows that there exists a permutation matrix $R_3$ of size $w \times w$ such that the southeastern $n_3 \times n_3$ submatrix of the matrix $\text{diag}\,(I_{n_1+w_+}, R_3, I_{n_3-w_-})M$ is nonsingular. The matrix $Q$=$\text{diag}(I_{n_1-w_+}, R_1, R_3, I_{n_3-w_-})M = PM$ satisfies the claims of Theorem 3.5.1.

## 3.6 Computing the determinant of a banded matrix based on a $3 \times 3$ block representation.

We may utilize the set $\Gamma_3(A)$ of section 3.3, and the equation (3.19) to recursively compute the determinant of any $n \times n$ matrix $M$ having bandwidth

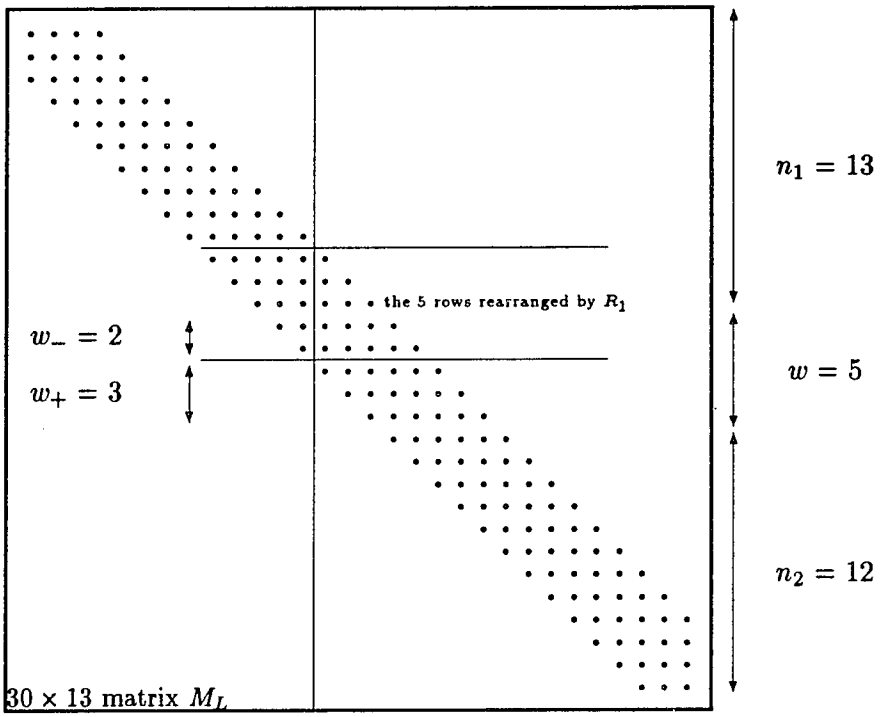



Figure 3.4: Matrix $M_L$ and positions of the rows rearranged by the permutation matrix $R_1$.

$w$ at an overall cost bounded by (3.3). Recall, however, that this bound is not optimal, since the Gaussian elimination algorithm supports the bound $O(nw^2)$ for this problem. Our goal, in this section, is to refine the algorithms of sections 3.3 and 3.4 for the problem $BAND \cdot LIN \cdot SOLVE$, so as to yield an optimum randomized algorithm for the problem $BAND \cdot DET$, which supports the bound (3.4). To this end, we first observe that only a subset of the scalars in the set $\Gamma_3(M)$ actually participates in the computation of det $M$. This subset consists of the entries to the $w \times w$ northwestern, northeastern,

southwestern and southeastern submatrices of the matrices $M^{-1}_{i_1 i_1 \ldots i_k i_k}$. Recall that the set $\Gamma_3(M)$ includes the first $w$ and the last $w$ columns of these matrices (which contains a substantially larger number of scalar parameters than the ones we actually need for the computation of det $M$). We will utilize the notations $N_W(M,n,w)$, $N_E(M,n,w)$, $S_W(M,n,w)$ and $S_E(M,n,w)$ for the $w \times w$ northwestern, northeastern, southwestern and southeastern submatrices of any $n \times n$ matrix $M$, respectively, where $n \geq w$. Proposition 3.2.1 implies the following auxiliary results.

FACT **3.6.1**

$$
\begin{aligned}
I_F(n,w) M I_F(n,w) &= \begin{pmatrix} N_W(M,n,w) & 0 \\ 0 & 0 \end{pmatrix}, \\
I_F(n,w) M I_L(n,w) &= \begin{pmatrix} 0 & N_E(M,n,w) \\ 0 & 0 \end{pmatrix}, \\
I_L(n,w) M I_F(n,w) &= \begin{pmatrix} 0 & 0 \\ S_W(M,n,w) & 0 \end{pmatrix}, \\
I_L(n,w) M I_L(n,w) &= \begin{pmatrix} 0 & 0 \\ 0 & S_E(M,n,w) \end{pmatrix}.
\end{aligned}
$$

### 3.6.1 Computing det $M$ (case where $M$ is strongly nonsingular).

THEOREM **3.6.1** *The computational cost of evaluating the determinant of an* $n \times n$ *strongly nonsingular matrix* $M$ *having bandwidth* $w$ *is bounded by (3.4).*

PROOF. Let $Z$ be the $w \times w$ matrix defined by (3.18), denote $M_{ij}$ the blocks of $M$ in representation-3, and define the set

$$\Gamma_3'(M) = \begin{cases} \{\det M,\ M^{-1}\}, & n < 3w, \\ \gamma_3'(M) \bigcup \Gamma_3'(M_{11}) \bigcup \Gamma_3'(M_{33}), & \text{otherwise}, \end{cases} \tag{3.55}$$

where

$$\begin{aligned} \gamma_2'(M) \ = \ \Big\{ & N_W(M^{-1}, n, w),\ N_E(M^{-1}, n, w), \\ & S_W(M^{-1}, n, w),\ S_E(M^{-1}, n, w),\ Z, \det M \Big\}. \end{aligned}$$

Since the set $\Gamma_3'(M)$ includes det $M$, it is sufficient to show that the computation of this set is bounded (3.4). To achieve the latter goal, expand (3.17), then premultiply and postmultiply both sides of the resulting matrix equation by $I_F(n,w)$ and $I_F(n,w)$, $I_F(n,w)$ and $I_L(n,w)$, $I_L(n,w)$ and $I_F(n,w)$, $I_L(n,w)$ and $I_L(n,w)$, respectively, then, apply Proposition 3.2.2 and Fact 3.6.1 and, finally, arrive at the four equations

$$\begin{aligned} N_W(M^{-1}, n, w) \ &= \ N_W(M_{11}^{-1}, n_1, w) + \\ & \qquad N_E(M_{11}^{-1}, n_1, w) M_{12}' Z^{-1} M_{21}' S_W(M_{11}^{-1}, n_1, w), \end{aligned} \tag{3.56}$$

$$S_W(M^{-1}, n, w) \ = \ S_W(M_{33}^{-1}, n_3, w) M_{32}' Z^{-1} M_{21}' S_W(M_{11}^{-1}, n_1, w), \tag{3.57}$$

$$N_E(M^{-1}, n, w) \ = \ N_E(M_{11}^{-1}, n_1, w) M_{12}' Z^{-1} M_{23}' N_E(M_{33}^{-1}, n_3, w), \tag{3.58}$$

$$\begin{aligned} S_E(M^{-1}, n, w) \ &= \ S_E(M_{33}^{-1}, n_3, w) + \\ & \qquad S_W(M_{33}^{-1}, n_3, w) M_{32}' Z^{-1} M_{23}' N_E(M_{33}^{-1}, n_3, w), \end{aligned} \tag{3.59}$$

where $M_{12}' = S_W(M_{12}, w)$, $M_{21}' = N_E(M_{21}, w)$, $M_{23}' = S_W(M_{23}, w)$ and $M_{32}' = N_E(M_{32}, w)$.

The set $\Gamma'_3(M)$ can be evaluated by performing the following computations:

ALGORITHM **3.6.1 (Determinant of a strongly nonsingular matrix).**

**input:** *an* $n \times n$ *strongly nonsingular matrix* $M$.

**output:** *the set* $\Gamma'_3(M)$ *of (3.55).*

**computations:**

1. *concurrently compute the sets* $\Gamma'_3(M_{11})$ *and* $\Gamma'_3(M_{33})$ *by recursively applying this algorithm, unless* $n \leq 3w$, *in which case we simply output* $\Gamma'_3(M) = \{\det M,\ M^{-1}\}$. Note that after this step, the matrices $N_W(M_{11}^{-1})$, $N_E(M_{11}^{-1})$, $S_W(M_{11}^{-1})$, $S_E(M_{11}^{-1})$, $N_W(M_{33}^{-1})$, $N_E(M_{33}^{-1})$, $S_W(M_{33}^{-1})$, $S_E(M_{33}^{-1})$ and the determinants $\det M_{11}$, $\det M_{33}$ are available;

2. *successively compute the matrix* $Z$ *[by applying (3.40)], at the cost*

$$O\left(t_{M(w,w,w)},\ p_{M(w,w,w)}\right), \tag{3.60}$$

*and the matrix* $Z^{-1}$ *at the cost*

$$O\left(t_{I(w)},\ p_{I(w)}\right); \tag{3.61}$$

3. *compute* $\det M$ *[by applying (3.19)] at a cost bounded by*

$$O\left(t_{D(w)},\ p_{D(w)}\right); \tag{3.62}$$

4. *compute the mairices* $N_W(M^{-1})$, $N_E(M^{-1})$, $S_W(M^{-1})$ *and* $S_E(M^{-1})$ *[by applying (3.56)-(3.59)] at a cost bounded by*

$$O\left(t_{M(w,w,w)},\ p_{M(w,w,w)}\right). \tag{3.63}$$

The equations (3.60)–(3.63) show that, once the sets $\Gamma'_3(M_{11})$ and $\Gamma'_3(M_{33})$ have been computed (stage 1 of Algorithm 3.6.1), the subsequent computation of the set $\gamma'_3(M)$, which consists of the remaining (unknown) elements of $\Gamma'_3(M)$), can be carried out at a cost bounded by

$$O\left(t_{I(w)} + t_{D(w)},\ p_{I(w)} + p_{D(w)}\right) \tag{3.64}$$

(this bound follows by combining the computational cost of stages 2, 3 and 4 of Algorithm 3.6.1).

Let $t' = t'(n,w)$ bound the number of parallel steps required for computing $\Gamma'_3(M)$ and let $p' = p'(n,w)$ bound the corresponding number of required processors. The above algorithm leads to the following relations for the pair $(t', p')$, estimated within a constant factor, where, for simplicity (with no loss of generality), we have assumed that $n = (2^q + q)w$, for some positive integer $q$:

$$t'(n,w) \leq \begin{cases} t_{I(w)}, & n < 2w; \\ t'(n/2,w) + t_{I(w)} + t_{D(w)}, & \text{otherwise;} \end{cases}$$

$$p'(n,w) \leq \begin{cases} p_{I(w)}, & n < 2w; \\ \max\left\{2\,p'(n/2,w),\ p_{I(w)} + p_{D(w)}\right\}, & \text{otherwise.} \end{cases}$$

A recursive application of the latter estimates and the B-principle yields the desired complexity bound (3.4). ■

### 3.6.2 Computing $\det M$ and $|\det M|$ (general case).

We now extend Algorithm 3.6.1 (see page 82) to computing the determinant of any banded matrix.

If the ground field is the field **C** of complex numbers, and, moreover, if we only need to compute $|\det M|$, we may use the symmetrization technique. The arguments are identical to the one of chapter 2, used there to relax the h.p.d. assumption about the input matrix, for a similar problem (see Remark 2.4.1, page §52).

In the case of any field, in order to relax the strong nonsingularity assumption for the problem $BAND \cdot DET$, we will again use randomization; specifically, we will shift from $M$ to $(PM)$ with $P = \text{diag}\,(I_{n_1-w/2}, R_1, R_2, I_{n_3-w/2})$, where $R_1$ and $R_2$ are random matrices having size $w \times w$ each. Due to the equation $\det M = (\det PM)/(\det P)$ and the bound $O\left(t_{I(w)}, p_{I(w)}\right)$ on the complexity of computing $\det P$, the cost of this transition does not increase the asymptotic computational complexity of the operations required in each parallel step of Algorithm 3.6.1 [see page 82, and recall that each step of Algorithm 3.6.1 has a combined cost bound $O\left(t_{I(w)} + t_{D(w)}, p_{I(w)} + p_{D(w)}\right)$]. Consequently, this transition does not increase the overall complexity bound (3.4) on solving the problem $BAND \cdot DET$. This algorithm can also detect singularity of the input matrix: we compute, as a by-product, the determi-

nant of all the matrices corresponding to $M_{i_1 i_1 \ldots i_k i_k}$; if all these determinants are nonzero, then, Algorithm 3.6.1 correctly computes det $M$; otherwise, at least one of these determinants is zero; in this case, we output det $M = 0$ and stop [the latter equation follows, due to the equation (3.19) and its recursive extension]. ∎

REMARK **3.6.1** If the matrix $M$ is nonsingular, the algorithm for computing the determinant can (wrongfully) output det $M = 0$ (though with a low probability). However, if the matrix $M$ is singular, this algorithm will always correctly output det $M = 0$.

# Chapter 4

## Solving a BLS over any field, based on a $2 \times 2$ block matrix representation.

In chapter 3, we presented a near optimal Las Vegas randomized algorithm for the problem $BAND \cdot LIN \cdot SOLVE$ and an optimal algorithm for the problem $BAND \cdot DET$, both over an arbitrary filed, via an improvement of the algorithm of [E]. In this chapter we present alternative solutions to these problems, which achieve the same results and, moreover, utilize fewer random parameters than the ones required by the algorithms of our previous chapter or by the algorithm of [E]. The solutions presented here rely on a $2 \times 2$ block matrix representation and on a factorization based on this representation, which appears to be simpler than the one of chapter 3. The algorithms of this chapter also support all the asymptotic bounds established in chapter 3, for similar problems, under similar conditions. We recall these bounds, applied to a nonsingular $n \times n$ input linear system that has the

bandwidth $w$.

For the problem $BAND \cdot LIN \cdot SOLVE$, we reach the Las Vegas randomized (respectively, deterministic) computational complexity cost bound

$$O\left(\left(\log \frac{n}{w}\right) t_{I(w)},\ \left(\frac{n}{w}\right) p_{I(w)}\right), \tag{4.1}$$

over any field of constants (respectively, any field of characteristic zero).

When we solve several linear systems, each having the same coefficient matrix $A$, we first compute a set of parameters denoted $\Gamma_2(A)$, at a cost bounded by (4.1); after that, we deterministically compute the solution of any linear system, having coefficient matrix $A$, at a cost bounded by

$$O\left(\left(\log \frac{n}{w}\right)(\log w), \frac{wn}{\log w}\right), \tag{4.2}$$

and we solve $s$ linear systems, each having the same coefficient matrix $A$, at a cost bounded by

$$O\left(\left(\log \frac{n}{w}\right)(\log w), \frac{swn}{\log w}\right). \tag{4.3}$$

Regarding the computation of the determinant, we reach the *optimum* Las Vegas randomized cost bound

$$O\left(\left(\log \frac{n}{w}\right)\left(t_{I(w)} + t_{D(w)}\right),\ \left(\frac{n}{w}\right) \frac{p_{I(w)} + p_{D(w)}}{\log \frac{n}{w}}\right), \tag{4.4}$$

for the problem $BAND \cdot DET$, over any field, and, if the input matrix is over the field $\mathbf{C}$ of complex numbers, we deterministically solve the problem $|BAND \cdot DET|$ (the problem of evaluating $|\det {}^2W|$), at a cost bounded by (4.4).

We organize our presentation in the following order: After some definitions and preliminary results in §4.1 and §4.2, we preprocess a strongly nonsingular matrix, in §4.3. We utilize the output of this preprocessing in §4.4 to effectively compute the solution of the input linear system. In §4.5, we relax the assumption about strong nonsingularity of the input matrix. In §4.6, we extend the solution of $BAND \cdot LIN \cdot SOLVE$ to the computation of the determinant of any banded matrix.

## 4.1 A factorization of a banded matrix based on a $2 \times 2$ block matrix representation.

Hereafter, $A$ denotes an $n \times n$ matrix having bandwidth $w$. We define *representation-2* of matrix $A$ (also referred to as to *balanced representation*) as the $2 \times 2$ block matrix

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \tag{4.5}$$

where $A_{11}$ and $A_{22}$ are $n_1 \times n_1$ and $n_2 \times n_2$ blocks, respectively, with

$$n_1 = \lceil n/2 \rceil, \quad n_2 = n - n_1. \tag{4.6}$$

$A_{i_1 i_1 \ldots i_k i_k}$ $(k \geq 0)$ denote the principal submatrices of $A$ specified by the relations

$$A_{i_1 i_1 \ldots i_k i_k} = \begin{cases} A, & k = 0, \\ (A_{i_1 i_1 \ldots i_{k-1} i_{k-1}})_{i_k i_k}, & k > 0; \end{cases} \tag{4.7}$$

where $i_1, \ldots, i_k \in \{1, 2\}$ and $(A_{i_1 i_1 \ldots i_{k-1} i_{k-1}})_{i_k i_k}$ denotes the $(i_k, i_k)$ block in representation-2 of matrix $A_{i_1 i_1 \ldots i_{k-1} i_{k-1}}$.

Until §4.5, we will assume that the matrix $A$ is strongly nonsingular. Consequently, all the submatrices $A_{i_1 i_1 \ldots i_k i_k}$ are nonsingular, and the matrices $A$ and $A^{-1}$ have the following factorizations, hereafter referred to as to *factorization-2* of matrix $A$:

$$A = \begin{pmatrix} I_{n_1} & 0 \\ A_{21}A_{11}^{-1} & I_{n_2} \end{pmatrix} \begin{pmatrix} A_{11} & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I_{n_1} & A_{11}^{-1}A_{12} \\ 0 & I_{n_2} \end{pmatrix}, \tag{4.8}$$

$$A^{-1} = \begin{pmatrix} I_{n_1} & -A_{11}^{-1}A_{12} \\ 0 & I_{n_2} \end{pmatrix} \begin{pmatrix} A_{11}^{-1} & 0 \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} I_{n_1} & 0 \\ -A_{21}A_{11}^{-1} & I_{n_2} \end{pmatrix}, \tag{4.9}$$

$$S = A_{22}Y, \tag{4.10}$$

$$Y = I_{n_2} - A_{22}^{-1}A_{21}A_{11}^{-1}A_{12}. \tag{4.11}$$

Here, $S$ denotes the customary *Schur complement* of $A_{11}$ in $A$. Note that the matrices $S$ and $Y$ are nonsingular, due to the equations (4.8) and (4.10) and to nonsingularity of the (1,1) and (2,2) blocks of $A$.

## 4.2 Some preliminary results.

As in the previous chapter, we will need to compute the first $w$ columns and the last $w$ columns of the matrices $A^{-1}_{i_1 i_1 \ldots i_k i_k}$. These columns will be expressed by utilizing the auxiliary matrices $I_F(n,w)$ and $I_L(n,w)$, introduced in the previous chapter [see (3.20), (3.21) and Proposition 3.2.1, page 64].

PROPOSITION **4.2.1** *Let $A$ be any $n \times n$ matrix having bandwidth $w$, where $n \geq 2w$. The (1,2) and (2,1) blocks in representation-2 of matrix $A$ satisfy*

*the relations*

$$A_{12} = I_L(n_1, w)A_{12} = A_{12}I_F(n_2, w) = I_L(n_1, w)A_{12}I_F(n_2, w); \qquad (4.12)$$

$$A_{21} = I_F(n_2, w)A_{21} = A_{21}I_L(n_1, w) = I_F(n_2, w)A_{21}I_L(n_1, w). \qquad (4.13)$$

These equations follow from Proposition 3.2.1 (see page 64), since the matrices $A_{12}$ and $A_{21}$ have the formats $\begin{pmatrix} 0 & 0 \\ L & 0 \end{pmatrix}$ and $\begin{pmatrix} 0 & U \\ 0 & 0 \end{pmatrix}$, respectively, where $U$ and $L$ are $w \times w$ matrices. ∎

PROPOSITION **4.2.2** *Once the first $w$ columns and the last $w$ columns of the matrices $A_{11}^{-1}$ and $A_{22}^{-1}$ have been computed, the matrix $Y$ of (4.11) can be computed at a cost bounded by*

$$O\left(t_{M(n,w,w)},\ p_{M(n,w,w)}\right) = O\left(\log w,\ \frac{nw^2}{\log w}\right), \qquad (4.14)$$

*and the matrix $Y^{-1}$ can be computed at a cost bounded by*

$$O\left(t_{I(w)},\ \left(\frac{n}{w}\right) p_{I(w)}\right). \qquad (4.15)$$

PROOF. Due to (4.12) and (4.13), we may substitute $I_L(n_1, w)A_{12}I_F(n_2, w)$ and $I_F(n_2, w)A_{21}I_L(n_1, w)$ for $A_{12}$ and $A_{21}$ in the equation (4.11), respectively, to derive the equation

$$Y = I_{n_2} - A_{22}^{-1}I_F(n_2, w)A_{21}I_L(n_1, w)A_{11}^{-1}I_L(n_1, w)A_{12}I_F(n_2, w). \qquad (4.16)$$

Since the first $w$ and the last $w$ columns of the matrices $A_{11}^{-1}$ and $A_{22}^{-1}$ are assumed computed, the matrices $A_{11}^{-1}I_F(n_1, w)$, $A_{11}^{-1}I_L(n_1, w)$, $A_{22}^{-1}I_F(n_2, w)$

and $A_{22}^{-1}I_L(n_2, w)$ are readily available; therefore, we may apply (4.16) to compute the matrix $Y$, at the price of a fixed constant number of $n_2 \times w$ by $w \times w$ matrix multiplications, which immediately implies the bound (4.14) on the complexity of computing the matrix $Y$. In order to compute $Y^{-1}$, apply Remark 3.2.1 (see page 65) to the equation (4.16) to conclude that the second term of the right hand side of the equation (4.16) has the format $(W \quad 0)$, where $W$ is a matrix of size $n_2 \times w$. It follows that $Y$ is a block triangular matrix of the format

$$Y = \begin{pmatrix} Y_{11} & 0 \\ Y_{21} & I_{n_2-w} \end{pmatrix}, \tag{4.17}$$

where $Y_{11}$ is a $w \times w$ matrix. Due to nonsingularity of $Y$ and to the latter equation (4.17), the matrix $Y_{11}$ is nonsingular; consequently,

$$Y^{-1} = \begin{pmatrix} Y_{11}^{-1} & 0 \\ -Y_{21}Y_{11}^{-1} & I_{n_2-w} \end{pmatrix}. \tag{4.18}$$

If the matrix $Y$ is available, then the latter equation enables us to compute $Y^{-1}$ at the price of a single inversion of a $w \times w$ matrix and a single multiplication of an $(n/2) \times w$ by a $w \times w$ matrices; these computations can be performed at a cost bounded by $O\left(t_{I(w)}, p_{I(w)}\right)$ and $O\left(t_{M(n,w,w)}, p_{M(n,w,w)}\right)$, respectively. Combine the two latter bounds and (4.14) (the bound on the cost of computing $Y$) with the B-principle, in order to derive the bound

$$O\left(t_{I(w)} + t_{M(n,w,w)}, \frac{t_{I(w)}p_{I(w)} + t_{M(n,w,w)}p_{M(n,w,w)}}{t_{I(w)} + t_{M(n,w,w)}}\right) = O\left(t_{I(w)}, \left(\frac{n}{w}\right)p_{I(w)}\right)$$

on the overall cost of computing $Y^{-1}$. Note that only the first $w$ columns of $Y$ and of $Y^{-1}$ need be computed. ∎

PROPOSITION **4.2.3** *Once the first $w$ and the last $w$ columns of the matrices $A_{11}^{-1}$ and $A_{22}^{-1}$ have been computed, the first $w$ and the last $w$ columns of the matrix $A^{-1}$ can be computed at a cost bounded by*

$$O\left(t_{I(w)}, \frac{n}{w} p_{I(w)}\right). \tag{4.19}$$

PROOF. Expand (4.9), then postmultiply both sides of the resulting equation by the matrix $I_F(n,w)$ [respectively, $I_L(n,w)$] and apply Proposition 4.2.1 to derive the two equations

$$\begin{aligned} A^{-1} I_F(n,w) &= \begin{pmatrix} A_{11}^{-1} I_F(n_1,w) + A_{11}^{-1} A_{12} Y^{-1} A_{22}^{-1} A_{21} A_{11}^{-1} I_F(n_1,w) & 0 \\ -Y^{-1} A_{22}^{-1} A_{21} A_{11}^{-1} I_F(n_1,w) & 0 \end{pmatrix} \\ &= \begin{pmatrix} F_1 + L_1 A_{12} Y^{-1} F_2 A_{21} F_1 & 0 \\ -Y^{-1} F_2 A_{21} F_1 & 0 \end{pmatrix}, \end{aligned} \tag{4.20}$$

and

$$\begin{aligned} A^{-1} I_L(n,w) &= \begin{pmatrix} 0 & -A_{11}^{-1} A_{12} Y^{-1} A_{22}^{-1} I_L(n_2,w) \\ 0 & Y^{-1} A_{22}^{-1} I_L(n_2,w) \end{pmatrix} \\ &= \begin{pmatrix} 0 & -L_1 A_{12} Y^{-1} L_2 \\ 0 & Y^{-1} L_2 \end{pmatrix}, \end{aligned} \tag{4.21}$$

respectively, where

$$F_1 = A_{11}^{-1} I_F(n_1,w), \tag{4.22}$$

$$L_1 = A_{11}^{-1} I_L(n_1,w), \tag{4.23}$$

$$F_2 = A_{22}^{-1} I_F(n_2, w), \tag{4.24}$$

$$L_2 = A_{22}^{-1} I_L(n_2, w). \tag{4.25}$$

The matrices $F_1$, $L_1$, $F_2$ and $L_2$ are readily available (since the first $w$ columns and the last $w$ columns of $A_{11}$ and $A_{22}$ are assumed computed), and, due to Proposition 4.2.2, the matrix $Y^{-1}$ can be computed at a cost bounded by (4.19). After that, due to the equations (4.20) and (4.21), the computation of $A^{-1}I_F(n,w)$ and $A^{-1}I_L(n,w)$ only requires a constant number of $(n/2) \times w$ by $w \times w$ matrix multiplications, which can be performed at a cost bounded by $O\left(t_{M(n,w,w)}, p_{M(n,w,w)}\right) = O\left(t_{I(w)}, \frac{n}{w} p_{I(w)}\right)$. ∎

## 4.3 Preprocessing a strongly nonsingular BLS, based on a $2 \times 2$ block representation of the input matrix.

Let the set of parameters computed by the preprocessing algorithm be denoted $\Gamma_2(A)$ and defined by the relation

$$\Gamma_2(A) = \begin{cases} \{A^{-1}\}, & n < 2w, \\ \gamma_2(A) \bigcup \Gamma_2(A_{11}) \bigcup \Gamma_2(A_{22}), & \text{otherwise;} \end{cases} \tag{4.26}$$

$$\gamma_2(A) = \left\{Y^{-1},\ A^{-1}I_F(n,w),\ A^{-1}I_L(n,w)\right\}.$$

Let $Y_{i_1 i_1 \ldots i_k i_k}$ denote the matrix corresponding to $Y$ in factorization-2 of $A_{i_1 i_1 \ldots i_k i_k}$ [note that the set $\Gamma_2(A)$ consists of the matrices $Y^{-1}_{i_1 i_1 \ldots i_k i_k}$ and the

submatrices of the matrices $A^{-1}_{i_1 i_1 \ldots i_k i_k}$ $(k \geq 0)$, formed by their first $w$ columns or their last $w$ columns].

We define the problem *PREPROCESS*-2 of a matrix $A$ (based on representation-2) as the computation of the set $\Gamma_2(A)$.

THEOREM **4.3.1** *The set $\Gamma_2(A)$, where $A$ is a strongly nonsingular $n \times n$ matrix $A$ having bandwidth $w$, can be computed at a cost bounded by (4.1).*

PROOF. Due to (4.26), in order to compute the set $\Gamma_2(A)$, we may first concurrently compute the sets $\Gamma_2(A_{11})$ and $\Gamma_2(A_{22})$; after that, the matrices $F_1$, $L_1$, $F_2$ and $L_2$ of (4.22)–(4.25) are available, and then, we may apply Proposition 4.2.2 (see page 90) and Proposition 4.2.3 (see page 92) to compute the remaining (unknown) elements of $\Gamma_2(A)$ [that is, the set $\gamma_2(A)$] at a cost bounded by $O\left(t_{I(w)}, \left(\frac{n}{w}\right) p_{I(w)}\right)$.

Let $t_1 = t_1(n, w)$ bound the number of parallel steps required for the overall computation of the set $\Gamma_2(A)$ and denote $p_1 = p_1(n, w)$ the corresponding bound on the number of required processors. The above algorithm leads to the following relations for the pair $(t_1, p_1)$, estimated within a constant factor (for simplicity, we may assume that $n = 2^q w$ for some positive integer $q$):

$$t_1(n,w) \leq \begin{cases} t_{I(w)}, & n < 2w; \\ t_1(n/2, w) + t_{I(w)}, & \text{otherwise}; \end{cases}$$

$$p_1(n,w) \leq \begin{cases} p_{I(w)}, & n < 2w; \\ \max\left\{2p_1(n/2, w), \left(\frac{n}{w}\right) p_{I(w)}\right\}, & \text{otherwise.} \end{cases}$$

Recursive application of the latter estimates yields the desired complexity bound (4.1) on the pair $(p_1, t_1)$. ∎

## 4.4 Solving a preprocessed BLS: $BACK \cdot SOLVE$-2.

We define the problem $BACK \cdot SOLVE$-2 as the computation of the vector $A^{-1}\overrightarrow{\mathbf{b}}$, for any vector $\overrightarrow{\mathbf{b}}$, given the set $\Gamma_2(A)$.

THEOREM **4.4.1** *Let $A$ be a strongly nonsingular matrix having bandwidth $w$, and let $\overrightarrow{\mathbf{b}}$ be any $n$-dimensional vector. Once the set $\Gamma_2(A)$ has been computed, the vector $A^{-1}\overrightarrow{\mathbf{b}}$ can be computed at a cost bounded by (4.2).*

PROOF. Let $\overrightarrow{\mathbf{b}} = \begin{pmatrix} \overrightarrow{\mathbf{b}_1} \\ \overrightarrow{\mathbf{b}_2} \end{pmatrix}$, where $\overrightarrow{\mathbf{b}_1}$ and $\overrightarrow{\mathbf{b}_2}$ are two vectors of dimensions $n_1$ and $n_2$, respectively. Expand (4.9), then postmultiply both sides of the resulting equation by the vector $\overrightarrow{\mathbf{b}}$ and apply Proposition 4.2.1 to derive the following equations where the matrices $L_1$ and $F_2$ are defined by (4.23) and (4.24), respectively:

$$A^{-1}\overrightarrow{\mathbf{b}} = \begin{pmatrix} A_{11}^{-1}\overrightarrow{\mathbf{b}_1} + A_{11}^{-1}A_{12}Y^{-1}A_{22}^{-1}A_{21}A_{11}^{-1}\overrightarrow{\mathbf{b}_1} - A_{11}^{-1}A_{12}Y^{-1}A_{22}^{-1}\overrightarrow{\mathbf{b}_2} \\ -Y^{-1}A_{22}^{-1}A_{21}A_{11}^{-1}\overrightarrow{\mathbf{b}_1} + Y^{-1}A_{22}^{-1}\overrightarrow{\mathbf{b}_2} \end{pmatrix}$$

$$= \begin{pmatrix} A_{11}^{-1}\overrightarrow{\mathbf{b}_1} + L_1A_{12}Y^{-1}F_2A_{21}A_{11}^{-1}\overrightarrow{\mathbf{b}_1} - L_1A_{12}Y^{-1}A_{22}^{-1}\overrightarrow{\mathbf{b}_2} \\ -Y^{-1}F_2A_{21}A_{11}^{-1}\overrightarrow{\mathbf{b}_1} + Y^{-1}A_{22}^{-1}\overrightarrow{\mathbf{b}_2} \end{pmatrix} \quad (4.27)$$

In order to compute $A^{-1}\overrightarrow{\mathbf{b}}$, first concurrently compute the two vectors $A_{11}^{-1}\overrightarrow{\mathbf{b}_1}$ and $A_{22}^{-1}\overrightarrow{\mathbf{b}_2}$ and, after that, apply (4.27) to compute the vector $A^{-1}\overrightarrow{\mathbf{b}}$

at the price of a fixed constant number of multiplications of matrices of size at most $w \times \lceil n/2 \rceil$ by vectors of dimension at most $\lceil n/2 \rceil$ [note that the matrices $Y^{-1}$, $L_1$ and $F_2$ are available from the set $\Gamma_2(A)$].

Let $t_2 = t_2(n,w)$ bound the number of parallel steps required for the computation of $A^{-1}\overrightarrow{\mathbf{b}}$ [with the assumption that $\Gamma_2(A)$ is available], and let $p_2 = p_2(n,w)$ denote the corresponding number of required processors. The above algorithm leads to the following relations for the pair $(t_2, p_2)$, estimated within a constant factor:

$$t_2(n,w) \leq \begin{cases} t_{M(w,w,1)}, & n \leq 2w, \\ t_2(n/2,w) + t_{M(n/2,w,1)}, & \text{otherwise;} \end{cases}$$

$$p_2(n,w) \leq \begin{cases} p_{M(w,w,1)}, & n \leq 2w, \\ \max\left\{2p_2(n/2,w),\ p_{M(n/2,w,1)}\right\}, & \text{otherwise.} \end{cases}$$

Recursive application of the latter relations yields the desired complexity bound (4.2). ∎

## 4.5 Relaxing the strong nonsingularity assumption.

Over the fields of characteristic zero, we may apply the symmetrization technique in order to relax the assumption about strong nonsingularity of the input matrix $M$, by shifting from $M$ to $M^H M$. The argument here is the same as the one of chapter 2 (see §2.3, page 46).

Alternatively, in the case of any field, we will again utilize an argument based on randomization. Following the line of §3.5 (see page 73), we will

assume (with no loss of generality) that the input matrix, hereafter denoted $A'$, is an $n \times n$ nonsingular matrix of the format

$$A' = A + \text{diag}\,(B, 0, C)\,, \tag{4.28}$$

where $w(A) = w$, $w_-(A) = w_+(A) = (w/2)$, and $B$, $C$ are $w \times w$ matrices (note that the matrices $A'$ correspond to $M'$ of §3.5). Denote $w' = w(A')$ (note that $w < w' < 2w$), $\mathcal{A}'(n, w)$ the set of matrices $A'$ satisfying the conditions listed above [note that $\mathcal{A}'(n, w)$ includes all nonsingular matrices having bandwidth $w$].

PROPOSITION **4.5.1** *Let* $P = \text{diag}\left(I_{n_1 - w/2}, R, I_{n_2 - w/2}\right)$, *where* $R$ *is a nonsingular matrix of size* $w \times w$ *and let* $A' \in \mathcal{A}'(n, w)$. *Denote* $(PA')_{ij}$ *the* $(i, j)$ *block in representation-2 of matrix* $PA'$. *The following properties hold:*

1. *the blocks* $(PA')_{12}$ *and* $(PA')_{21}$ *have the formats* $\begin{pmatrix} 0 & 0 \\ L & 0 \end{pmatrix}$ *and* $\begin{pmatrix} 0 & U \\ 0 & 0 \end{pmatrix}$, *respectively, where* $L$ *and* $U$ *are* $w \times w$ *matrices. Consequently, Proposition 4.2.1 (see page 89) still holds if we substitute* $(PA')_{ij}$ *for* $A_{ij}$ $(i, j \in \{1, 2\},\ i \neq j)$;

2. *if* $(PA')_{11}$ *and* $(PA')_{22}$ *are nonsingular, then* $(PA')_{11} \in \mathcal{A}'(n_1, w)$ *and* $(PA')_{22} \in \mathcal{A}'(n_2, w)$.

This proposition follows, by a simple inspection of the product $PA'$ (see Figure 4.1). ∎

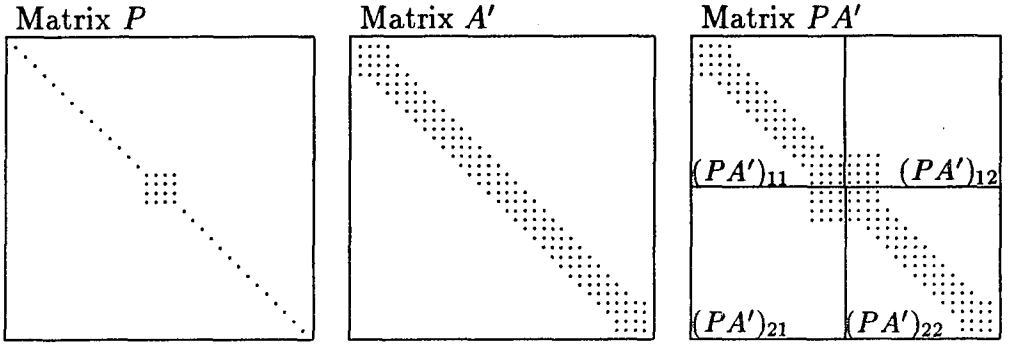



Figure 4.1: Format of the matrices $(PA')_{ij}$, where $A' \in \mathcal{A}(32,4)$, $P = \text{diag}\,(I_{14}, R, I_{14})$; here, $R$ is a $4 \times 4$ matrix.

---

We now show how to transform any linear system of the format

$$A'\overrightarrow{\mathbf{x}} = \overrightarrow{\mathbf{b}}, \tag{4.29}$$

where $A' \in \mathcal{A}'(n,w)$, into a suitable input to the preprocessing algorithm of section 4.3. Towards this goal, we will shift from (4.29) to

$$PA'\overrightarrow{\mathbf{x}} = P\overrightarrow{\mathbf{b}}, \tag{4.30}$$

with

$$P = \text{diag}\,\left(I_{n_1-w/2}, R, I_{n_2-w/2}\right), \tag{4.31}$$

where $R$ is a random matrix of size $w \times w$. Since the matrix $P$ is nonsingular, the equations (4.29) and (4.30) are equivalent; therefore, the vector $A'^{-1}\overrightarrow{\mathbf{b}}$ is obtained by simply computing $(PA')^{-1}(P\overrightarrow{\mathbf{b}})$. The transition from (4.29) to (4.30) amounts to a single $w \times w$ matrix-by-matrix multiplication and a single

$w \times w$ matrix-by-vector multiplication. The bound $O\left(t_{M(w,w,w)}, p_{M(w,w,w)}\right)$ on the computational cost of this transition does not increase the bound (4.19) on the computational complexity of each parallel step of the preprocessing algorithm of section 3.3. In order to verify that the auxiliary system (4.30) is a well suited input to this preprocessing algorithm, we only need to ensure the two following conditions:

a) The (1,2) and (2,1) blocks in representation-2 of matrix $PA'$ should verify Proposition 4.2.1 (see page 89). This condition immediately follows from Proposition 4.5.1 (see page 97).

b) The matrices $PA'$, $(PA')_{11}$, $(PA')_{22}$ and $Y_{PA'}$ should be nonsingular (here, $Y_{PA'}$ denotes the matrix corresponding to $Y$ in factorization-2 of matrix $PA'$), but since $PA'$ is nonsingular (with a high probability) and since nonsingularity of $Y_{PA'}$ follows from nonsingularity of $PA'$, $(PA)'_{11}$ and $(PA)'_{22}$, it is sufficient to ensure nonsingularity of $(PA')_{11}$, $(PA')_{22}$, assuming that $PA'$ is nonsingular. We show in Theorem 4.5.1 (see page 100) how to assign the entries of $R$ so as to yield nonsingularity for $(PA')_{11}$ or $(PA')_{22}$. By the standard argument of [Sc], [Zi], nonsingularity of these two principal submatrices of $PA'$ follows (with a high probability) for a random assignment of values (from a fixed large set) to the entries of $R$.

Moreover, Proposition 4.5.1 shows that the matrices $(PA')_{11}$ and $(PA')_{22}$

have the format of the original input matrix $A'$, and, consequently, the randomization process can be continued recursively. ∎

THEOREM **4.5.1** *Let* $A' = A + \text{diag}\ (B, 0, C)$ *be a any matrix in* $\mathcal{A}'(n, w)$ *where* $n > 2w$, *and let* $n_1$, $n_2$ *be defined by (4.6). Then there exists a permutation matrix of the format* $P = \text{diag}\left(I_{n_1 - w_+(A)}, R, I_{n_2 - w_-(A)}\right)$ *where* $R$ *is a* $w \times w$ *matrix, such that the block* $(PA')_{11}$ *in representation-2 of the matrix* $PA'$ *is nonsingular.*

PROOF. Let $A'_L$ denote the $n \times n_1$ matrix $\begin{pmatrix} A'_{11} \\ A'_{21} \end{pmatrix}$, where $A'_{11}$ and $A'_{22}$ are the block (1,1) and (2,1) (in representation-2) of $A'$. Assume (without loss of generality) that $w_+(A) = w_-(A) = k$. Since $A'$ is nonsingular, $A'_L$ has full (column) rank $n_1$. The top $n_1 - k$ rows of $A'_L$ are linearly independent, since the top $n_1 - k$ rows of $A'$ are linearly independent and since these rows have no nonzero entries outside $A'_L$. The top $n_1 + k$ rows of $A'_L$ have full rank, $n_1$, as well, because $A'_L$ has this rank and has no nonzero entries below these rows. It follows that there exists a set of $n_1$ rows of $A'_L$ that includes the first $n_1 - k$ top rows, as well as $k$ of the next $2k = w$ rows, that form a nonsingular $n_1 \times n_1$ matrix. Consequently, there exists a permutation matrix $R$ of size $2k \times 2k$ such that the leading principal $n_1 \times n_1$ submatrix of the matrix $PA' = \text{diag}\ (I_{n_1 - k}, R, I_{n_2 - k})\, A'$ is nonsingular. ∎

Similarly, nonsingularity of $(PA')_{22}$ (for a certain assignment of the random entries of $R$) follows. By the standard argument of [Sc], [Zi], nonsingularity of $B_{11}$ and of $B_{22}$ then follows (with a high probability) for a random

assignment of the values (from a fixed large set) to the entries of $R$.

REMARK **4.5.1** The overall number of parameters used in order to achieve the entire recursive factorization is $O(w^2(1+2+2^2+\ldots+2^{\log(n/w)}) = O(nw)$.

REMARK **4.5.2 (Solving several BLS's).** The algorithms of §4.4 for the problem $BAND \cdot LIN \cdot SOLVE$ can be extended to solving several linear systems of the format

$$A\vec{\mathbf{x}}(i) = \vec{\mathbf{b}}(i), \quad i = 1, \ldots, s, \tag{4.32}$$

at a cost bounded by (4.3) [assuming available the set $\Gamma_2(A)$]. In order to reach this bound, we substitute two $n \times s$ matrices denoted $X$ and $B$ for the vectors $\vec{\mathbf{x}}$ and $\vec{\mathbf{b}}$, respectively, in the proof of Theorem 4.4.1 [here, $B = \left(\vec{\mathbf{b}}_1, \ldots, \vec{\mathbf{b}}_s\right)$].

REMARK **4.5.3 (On the storage space).** It is not necessary to store the set $\Gamma_2(A)$, since each parallel step of the preprocessing and backsolving algorithms only involves the elements of $\Gamma_2(A)$ computed in the previous step. Consequently, the results of the computations performed by the current step can overwrite the ones of the previous step, so as to keep the storage space of these algorithms bounded by $O(wn)$ in the case of a single linear system and by $O(nw + ns)$ in the case of $s$ linear systems each having the same coefficient matrix. Note that this space bound on the solution of several linear systems applies if we are given $s$ distinct vectors $\vec{\mathbf{b}}(h)$, $h = 1, \ldots, s$, and are

required to solve the $s$ linear systems $A\vec{\mathbf{x}}(h) = \vec{\mathbf{b}}(h)$. However, if the vector $\vec{\mathbf{b}}(h+1)$ becomes available only after the system $A\vec{\mathbf{x}}(h) = \vec{\mathbf{b}}(h)$ has been solved, then, in order to solve each of the linear systems $A\vec{\mathbf{x}}(h) = \vec{\mathbf{b}}(h)$ at a cost bounded by (4.2) after a single preprocessing of matrix $A$, we will need to store the set $\Gamma_2(A)$. This additional space will increase the storage space of the solution to $O(nw \log \frac{n}{w})$.

## 4.6 Computing the determinant of a banded matrix.

THEOREM **4.6.1** *The computation of* det $A$, *for any* $n \times n$ *matrix* $A$ *having bandwidth* $w$, *can be performed at a cost bounded by (4.4).*

PROOF. We will assume that the matrix $A$ is strongly nonsingular (this assumption can be relaxed by applying Remark 4.6.1, see below). Then, we may utilize factorization-2 of matrix $A$ to derive the equation

$$\det A = \det A_{11} \det A_{22} \det Y. \tag{4.33}$$

Due to (4.17), $\det Y = \det Y_{11}$, and by combining the two latter equations , we obtain the equation

$$\det A = \det A_{11} \det A_{22} \det Y_{11}. \tag{4.34}$$

We now present an algorithm for computing det $A$, which supports the bound (4.4); this algorithm is based on the above equation (4.34) and utilizes a technique similar to the one of chapter 3 (see §3.6, page 78); specifically, we

will compute a set, hereafter denoted $\Gamma_2'(A)$ and defined by the recurrence relation

$$\Gamma_2'(A) = \begin{cases} \{\det A,\, A^{-1}\}, & n < 2w, \\ \gamma_2'(A) \cup \Gamma_2'(A_{11}) \cup \Gamma_2'(A_{22}), & \text{otherwise}, \end{cases} \tag{4.35}$$

where

$$\gamma_2'(A) = \{N_W(A^{-1}, n, w),\ N_E(A^{-1}, n, w),\ S_W(A^{-1}, n, w),\ S_E(A^{-1}, n, w),\ Y_{11},\ \det A\}.$$

Here, $N_W(X, n, w), N_E(X, n, w), S_W(X, n, w)$ and $S_E(X, n, w)$ denote the $w \times w$ northwestern, northeastern, southwestern and southeastern submatrices of any $n \times n$ matrix $X$, respectively (compare the set $\Gamma_2'(A)$ with the set $\Gamma_3'(A)$ of chapter 3, page 81). Since $\det A$ is a member of the set $\Gamma_2'(A)$, it is sufficient to show that the cost of computing this set is bounded by (4.4), in order to prove Theorem 4.6.1. Due to (4.35), the computation of $\Gamma_2'(A)$ can be performed as follows: we first concurrently compute the sets $\Gamma_2'(A_{11})$ and $\Gamma_2'(A_{22})$, then, we compute the small set $\gamma_2'(A)$ at a cost bounded by $O\left(t_{I(w)} + t_{D(w)}, p_{I(w)} + p_{D(w)}\right)$ (see Theorem 3.6.1, page 80, for a similar and more detailed proof). We may now deduce the bound (4.4) by recursively applying the above argument and by using the B-principle. ∎

REMARK 4.6.1 **(Relaxing the strong nonsingularity assumption).** Shifting from $A$ to $PA$ (as in §4.5) enables us to relax the assumption about strong nonsingularity of $A$ [recall that $P = \text{diag}\,(I_{n_1 - w/2}, R, I_{n_2 - w/2})$, where $R$ is a random matrix of size $w \times w$]. Due to the equation $\det A = (\det PA)/(\det P)$ and to the bound $O\left(t_{D(w)}, p_{D(w)}\right)$ on the cost of computing $\det P$, the price

of this transition does not increase the bound

$$O\left(t_{I(w)} + t_{D(w)},\ p_{I(w)} + p_{D(w)}\right)$$

on the cost on the computations required in each parallel step of the solution of the problem $BAND \cdot DET$ in the strongly nonsingular case.

REMARK **4.6.2** If the matrix $A$ is defined over the field $\mathbf{C}$ of complex numbers, the problem $|BAND \cdot DET|$ can be solved deterministically at a cost bounded by (4.4), since we may relax the strong nonsingularity assumption by means of symmetrization (see chapter 2, §2.4).

# Chapter 5

## Banded linear systems with upper and/or lower edge(s).

We present algorithms for a special case of the problems $BAND \cdot LIN \cdot SOLVE$, $BAND \cdot DET$, $PREPROCESS$ and $BACK \cdot SOLVE$, denoted $BAND \cdot LIN \cdot SOLVE^{\star}$, $BAND \cdot DET^{\star}$, $PREPROCESS^{\star}$ and $BACK \cdot SOLVE^{\star}$, respectively, where the input matrix has lower and/or upper edge(s). This requirement holds for a large class of banded linear systems; in particular, it typically holds for the matrices encountered in applications to $PDE's$ and $ODE's$ (see [Am] and [LP]).

Assuming an $n \times n$ nonsingular banded input matrix $A$ having bandwidth $w$ and having lower and/or upper edge(s), we solve the problem $BAND \cdot LIN \cdot SOLVE^{\star}$ at an overall *deterministic* computational cost bounded by

$$O\left(\left(\log \frac{n}{w}\right)(\log w) + t_{I(w)},\ \frac{\left(\frac{n}{w}\right) t_{I(w)}\, p_{I(w)}}{t_{I(w)} + \left(\log \frac{n}{w}\right)(\log w)}\right). \tag{5.1}$$

The estimates (5.1) also applies to the problem $BAND \cdot LIN \cdot SOLVE$ when the input matrix is block bidiagonal or banded triangular.

Moreover, when we solve several linear systems, each having the same coefficient matrix $A$, we preprocess $A$ at a deterministic cost bounded by (5.1), and, after that, we solve any linear system, having coefficient matrix $A$, at a deterministic cost bounded by

$$O\left(\left(\log \frac{n}{w}\right)(\log w),\ \frac{wn}{\left(\log \frac{n}{w}\right)(\log w)}\right), \tag{5.2}$$

which immediately implies the deterministic cost bound

$$O\left(\left(\log \frac{n}{w}\right)(\log w),\ \frac{qwn}{\left(\log \frac{n}{w}\right)(\log w)}\right) \tag{5.3}$$

on solving $q$ such linear systems.

The bounds (5.1), (5.2) and (5.3) also apply to their corresponding problems, respectively, when the input matrix is block bidiagonal (which includes the banded triangular case).

Furthermore, we solve the problem $BAND\text{-}DET^{\star}$ at a deterministic and optimal cost bounded by

$$O\left(\left(\log \frac{n}{w}\right)(\log w) + t_{I(w)} + t_{D(w)},\ \frac{\left(\frac{n}{w}\right)\left(t_{I(w)}\, p_{I(w)} + t_{D(w)}\, p_{D(w)}\right)}{t_{I(w)} + t_{D(w)} + \left(\log \frac{n}{w}\right)(\log w)}\right). \tag{5.4}$$

Regarding the storage space requirements, we need, besides the storage space $O(nw)$, required for the input linear system, an extra space for the auxiliary parameters computed by the preprocessing algorithm. This extra space, however, is no more than the one needed for the input linear system. It might be worth recalling that the computation of the inverse of a banded matrix, for the purpose of solving associated linear systems, is not a viable

approach, since this inverse is generally a full matrix, which requires $O(n^2)$ space, for an $n \times n$ banded matrix having bandwidth $w$. Such a bound is too large compared to only $O(nw)$ space needed for the input linear system.

The main idea behind the techniques of this chapter is to reduce the problem $BAND \cdot LIN \cdot SOLVE^{\star}$ to the problem of solving a nonsingular block bidiagonal linear system, which we then solve by means of a variant of the block cyclic reduction algorithm.

We organize the sections of this chapter in the following order: In the next four sections, we present a variant of the block cyclic reduction algorithm applied to a nonsingular block bidiagonal linear system and show how this technique can be combined with preprocessing to accelerate the computation of the solution of several nonsingular block bidiagonal linear systems, each having the same coefficient matrix. In §5.5, we reduce the problem of solving any nonsingular banded linear system having lower and/or upper edge(s) to the block bidiagonal case. In §5.6, we extend our algorithm to solving the problem $BAND \cdot DET^{\star}$.

## 5.1 Solving a nonsingular block bidiagonal linear system (with unit diagonal blocks).

Hereafter, $U$ denotes a nonsingular $k \times k$ block bidiagonal matrix with $w \times w$ blocks of the format

$$U = \begin{pmatrix} I_w & B_1 & 0 & \dots & & 0 \\ 0 & I_w & B_2 & \dots & & 0 \\ 0 & 0 & I_w & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \ddots & \\ 0 & 0 & \dots & & I_w & B_{k-1} \\ 0 & 0 & \dots & & 0 & I_w \end{pmatrix}. \tag{5.5}$$

Throughout this chapter, we will assume (with no loss of generality) that $k = 2^r$, for some positive integer $r$.

THEOREM **5.1.1** *The computational complexity of solving the linear system*

$$U\vec{\mathbf{x}} = \vec{\mathbf{b}} \tag{5.6}$$

*is bounded by*

$$O\left((\log k)(\log w),\ \frac{k\,w^3}{(\log k)(\log w)}\right), \tag{5.7}$$

*which turns into the equation*

$$O\left(\left(\log \frac{n}{w}\right)(\log w),\ \frac{n\,w^2}{\left(\log \frac{n}{w}\right)(\log w)}\right), \quad for\ k = \frac{n}{w}. \tag{5.8}$$

PROOF. Let $S^{(r)}$ and $U^{(r)}$ denote the input linear system and its coefficient matrix $U$, respectively. Set $B_{2^r} = 0$ and rewrite $S^{(r)}$ as a set of $2^r$ vector equations of the format

$$\vec{\mathbf{x}}_h + B_{h+1}\vec{\mathbf{x}}_{h+1} = \vec{\mathbf{b}}_h, \quad 0 \le h \le 2^r - 1, \tag{5.9}$$

where $\vec{x}_i$ and $\vec{b}_i$ are $w$-dimensional column subvectors of the vectors

$$\vec{x} = \left(\vec{x}_0^{\,T}, \ldots, \vec{x}_k^{\,T}\right)^T \quad \text{and} \quad \vec{b} = \left(\vec{b}_0^{\,T}, \ldots, \vec{b}_k^{\,T}\right)^T,$$

respectively. Eliminate the odd-indexed vector variables of $S^{(r)}$ to derive a smaller linear system [denoted $S^{(r-1)}$] by applying the following algorithm on $S^{(r)}$.

ALGORITHM **5.1.1 (BCRU-reduce).**

**Input:** *a block bidiagonal linear system denoted $S^{(l)}$ and consisting of $2^l$ vector equations of the format (5.9) where $B_{2^l} = 0$.*

**Output:** *a block bidiagonal linear system denoted $S^{(l-1)}$, consisting of $2^{l-1}$ vector equations of the format (5.9) as well, and whose solution determines all the even-indexed vector variables of $S^{(l)}$.*

**Computations:**

**for** $i := 0$ **to** $2^{l-1} - 1$ **do** (in parallel for all values of $i$)

1. *premultiply both sides of (5.9), for $h = 2i + 1$, by the matrix $(-B_{2i+1})$ to form a new vector equation denoted $E_1(i)$;*
2. *substitute $2i$ for $h$ in (5.9) to form a new equation denoted $E_2(i)$;*
3. *add $E_1(i)$ and $E_2(i)$ to form the output equation $E(i)$.* ■

The linear system $S^{(r-1)}$, output by Algorithm 5.1.1 on input $S^{(r)}$, has the format

$$\vec{\mathbf{x}}_{2h} - B_{2h+1}B_{2h+2}\vec{\mathbf{x}}_{2h+2} = \vec{\mathbf{b}}_{2h} - B_{2h+1}\vec{\mathbf{b}}_{2h+1}, \qquad 0 \leq h \leq 2^{r-1} - 1. \tag{5.10}$$

A simple inspection shows that the variables of $S^{(r-1)}$ consist exactly of the even-indexed vector variables of $S^{(r)}$. Since the equations of $S^{(r-1)}$ are linear combinations of the ones of $S^{(r)}$, the solution of $S^{(r-1)}$ partially solves $S^{(r)}$. Once $S^{(r-1)}$ is solved, each odd-indexed and yet noncomputed vector variable (say, $\vec{\mathbf{x}}_{2i+1}$) of $S^{(r)}$ is immediately computed by substituting $2i+1$ for $h$ in (5.9) and by replacing the vector $\vec{\mathbf{x}}_{2i+2}$ in the resulting equation by its value available from the solution of $S^{(r-1)}$. This computation of the vector $\vec{\mathbf{x}}_{2i+1}$ only amounts to a single multiplication of a $w \times w$ matrix by a $w$-dimensional vector plus a single $w$-dimensional vector addition. Moreover, the linear systems $S^{(r)}$ and $S^{(r-1)}$ have the same format (5.9), so that this reduction process can be continued recursively, by applying Algorithm 5.1.1 to $S^{(r-1)}$. A complete pseudo-code for this algorithm can be summarized as follows:

ALGORITHM **5.1.2 (BCRU).**

**Input:** *a block bidiagonal linear system* $U\vec{\mathbf{x}} = \vec{\mathbf{b}}$ *denoted* $S^{(l)}$ *and consisting of* $2^l$ *vector equations of the format (5.9) where* $B_{2^l} = 0$.

**Output:** *the vector* $\vec{\mathbf{x}} = \left(\vec{\mathbf{x}}_0, \ldots, \vec{\mathbf{x}}_{2^l-1}\right) = U^{-1}\vec{\mathbf{b}}$.

**Computations:**

**if** $l > 0$ **then**

1. *perform Algorithm 5.1.1 (BCRU-reduce) with input* $S^{(l)}$ *to compute* $S^{(l-1)}$ *at a cost bounded by* $O(\log w,\ (k/2)w^3)$;
2. *solve* $S^{(l-1)}$ *by a recursive application of the current algorithm to* $S^{(l-1)}$ *[after this step the values of the even-indexed vector variables of* $S^{(l)}$ *are available];*
3. *compute (in parallel) all the odd-indexed vector variables of* $S^{(l)}$ *(via back-substitution) at a cost bounded by* $O(\log w,\ (k/2)w^2)$.

Let $t = t(r, w)$ bound the number of parallel steps and let $p = p(r, w)$ bound the corresponding number of required processors, in the latter algorithm. The above arguments lead to the following relations for the pair $(t, p)$, estimated within multiplicative constant factors:

$$t(r,w) \leq \begin{cases} 0, & r = 0; \\ t(r-1,w) + t_{M(w,w,w)}, & \text{otherwise;} \end{cases}$$

$$p(r,w) \leq \begin{cases} 0, & r = 0; \\ \max\left\{p(r-1,w),\ 2^{r-1}p_{M(w,w,w)}\right\}, & \text{otherwise.} \end{cases}$$

Recursive application of the latter relations and the B-principle yield the desired complexity bound (5.7) for the pair $(t, p)$. ∎

## 5.2 Preprocessing a nonsingular block bidiagonal linear system (with unit diagonal blocks).

When we solve several linear systems of the format

$$U\vec{\mathbf{x}}(i) = \vec{\mathbf{b}}(i), \quad 1 \leq i \leq q, \tag{5.11}$$

we may substitute in Algorithm 5.1.2 two $n \times q$ matrices $X$ and $B$ for the vectors $\vec{\mathbf{x}}$ and $\vec{\mathbf{b}}$, respectively, with $B = \left(\vec{\mathbf{b}}(1), \ldots, \vec{\mathbf{b}}(q)\right)$. However, there are situations where this kind of substitution of a matrix for a vector is not appropriate. For instances, the vector $\vec{\mathbf{b}}(i+1)$ may become available only after the vector $\vec{\mathbf{x}}(i)$ has been determined, or the actual computer storage available may not be large enough to accommodate the matrix $B$. These situations can be dealt with by computing, at a preprocessing stage, a set of parameters, here denoted $\Gamma_{bbu}(U)$ and implicitly defining $U^{-1}$. Once available, this set can be utilized to achieve the same acceleration on the computation of the solution of (5.11), as if we had substituted the matrices $X$ and $B$ for the vectors $\vec{\mathbf{x}}$ and $\vec{\mathbf{b}}$ in Algorithm 5.1.2.

Hereafter, in addition to the notations introduced in the previous section, let $U^{(r)}$ and $U^{(r-1)}$ denote the coefficient matrices of $S^{(r)}$ and $S^{(r-1)}$, respectively, where $U^{(r)} = U$ and $S^{(r-1)}$ denotes the output of Algorithm 5.1.1 (BCRU-reduce, see page 109) on input $S^{(r)}$. Define the set $\Gamma_{bbu}(U)$ by the relation

$$\Gamma_{bbu}(U) = \Gamma_{bbu}(U^{(r)}) = \begin{cases} \emptyset, & \text{if } r = 0; \\ \{U^{(r-1)}\} \bigcup \Gamma_{bbu}(U^{(r-1)}), & \text{otherwise,} \end{cases} \tag{5.12}$$

and define *PREPROCESS*-bbu of matrix $U$ as the problem of computing the set $\Gamma_{bbu}(U)$.

THEOREM **5.2.1** *The problem PREPROCESS-bbu, for any $k \times k$ nonsingular block bidiagonal matrix $U$ with $w \times w$ blocks, of the format of (5.5), can be solved at a deterministic computational cost bounded by (5.7).*

PROOF. This bound follows, since all the elements of the set $\Gamma_{bbu}(U)$ are computed in the proof of Theorem 5.1.1 (see page 108). ∎

## 5.3 Solving a preprocessed block bidiagonal linear system.

THEOREM **5.3.1** *Once the set $\Gamma_{bbu}(U)$ has been computed, the linear system $U\vec{x} = \vec{b}$ can be solved, deterministically, at a cost bounded by*

$$O\left((\log k)\log w,\ \frac{kw^2}{(\log k)(\log w)}\right), \tag{5.13}$$

*which turns into the equation*

$$O\left(\left(\log\frac{n}{w}\right)\log w,\ \frac{nw}{\left(\log\frac{n}{w}\right)(\log w)}\right),\quad for\ k=\frac{n}{w}. \tag{5.14}$$

PROOF. Given $S^{(r)}$, we only need to compute the right hand side of each of the $k/2$ vector equations of $S^{(r-1)}$ [see (5.10)], in order to obtain the auxiliary linear system $S^{(r-1)}$ [since the coefficient matrix of $S^{(r-1)}$ is available from the set $\Gamma_{bbu}(U)$]. This computation amounts to $k/2$ concurrent multiplications of $w \times w$ matrices by $w$-dimensional vectors, which can be performed at a

cost bounded by $O(t_{M(w,w,1)}, kp_{M(w,w,1)})$. Moreover, once $S^{(r-1)}$ is solved, the odd-indexed variables of $S^{(r)}$ can be computed at a similar cost. By recursively applying this argument and using the B-principle, we arrive at the bound (5.13) on the cost of solving the system $U\vec{\mathbf{x}} = \vec{\mathbf{b}}$. ∎

## 5.4 Solving a nonsingular block bidiagonal linear system (general case).

Hereafter, $T$ denotes a nonsingular $k \times k$ block bidiagonal matrix with $w \times w$ blocks, of the format

$$T = \begin{pmatrix} A_0 & B_1 & 0 & \dots & & 0 \\ 0 & A_1 & B_2 & \dots & & 0 \\ 0 & 0 & A_2 & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \ddots & \\ 0 & 0 & \dots & & A_{k-2} & B_{k-1} \\ 0 & 0 & \dots & & 0 & A_{k-1} \end{pmatrix}. \tag{5.15}$$

Denote $D = \text{diag}\,(A_0, \dots, A_k)$ and $U' = D^{-1}T$. Observe that the matrix $U'$ has the format of the matrix $U$ defined by (5.5). Define the set

$$\Gamma_{bbt}(T) = \{D^{-1}, U'\} \bigcup \Gamma_{bbu}(U'). \tag{5.16}$$

Define *PREPROCESS*-bbt (of matrix $T$) as the problem of computing the set $\Gamma_{bbt}(T)$.

THEOREM **5.4.1** *Let $T$ be any matrix of the format (5.15).*

1. *The set* $\Gamma_{bbt}(T)$ *can be computed, deterministically, at a cost bounded by*

$$O\left((\log k)(\log w) + t_{I(w)},\ \frac{kt_{I(w)}\, p_{I(w)}}{t_{I(w)} + (\log k)(\log w)}\right), \tag{5.17}$$

*which turns into the equation*

$$O\left(\left(\log \frac{n}{w}\right)(\log w) + t_{I(w)},\ \frac{\left(\frac{n}{w}\right) t_{I(w)}\, p_{I(w)}}{t_{I(w)} + \left(\log \frac{n}{w}\right)(\log w)}\right), \tag{5.18}$$

*for* $k = \frac{n}{w}$.

2. *Once the set* $\Gamma_{bbt}(T)$ *has been computed, the linear system* $T\vec{\mathbf{x}} = \vec{\mathbf{b}}$ *can be solved, deterministically, at a cost bounded by*

$$O\left((\log k)(\log w),\ \frac{kw^2}{(\log k)(\log w)}\right), \tag{5.19}$$

*which turns into the equation*

$$O\left(\left(\log \frac{n}{w}\right)(\log w),\ \frac{wn}{\left(\log \frac{n}{w}\right)(\log w)}\right), \quad \text{for } k = \frac{n}{w}. \tag{5.20}$$

PROOF. The set $\Gamma_{bbt}(T)$ can be obtained by performing the following computations:

ALGORITHM **5.4.1 (Preprocessing-bbt).**

1. *concurrently compute the matrices* $A_i^{-1}$, *for* $1 \leq i \leq k$, *to obtain the matrix* $D^{-1}$ *at a cost bounded by* $O(t_{I(w)},\ kp_{I(w)})$;

2. *compute the matrix* $U' = D^{-1}T$ *at a cost bounded by*

$$O(t_{M(w,w,w)},\ kp_{M(w,w,w)});$$

3. *compute the set* $\Gamma_{bbu}(U')$, *by applying Theorem 5.2.1 (see page 113), at a cost bounded by*

$$O\left((\log k)(\log w),\ \frac{k\,w^3}{(\log k)(\log w)}\right).$$

Combine the three latter bounds to derive the bound (5.17) on the complexity of computing the set $\Gamma_{bbt}(T)$.

We now extend preprocessing-bbt to computing the solution to the linear system $T\overrightarrow{\mathbf{x}} = \overrightarrow{\mathbf{b}}$. Once the set $\Gamma_{bbt}(T)$ has been computed, then, due to the equation $T^{-1} = (D^{-1}T)^{-1}D^{-1}$, the solution vector $T^{-1}\overrightarrow{\mathbf{b}}$ is obtained by computing $U'^{-1}\overrightarrow{\mathbf{b}}'$, where $U' = D^{-1}T$ and $\overrightarrow{\mathbf{b}}' = D^{-1}\overrightarrow{\mathbf{b}}$. Note that the transition from $T^{-1}\overrightarrow{\mathbf{b}}$ to $U'^{-1}\overrightarrow{\mathbf{b}}'$ only amounts to computing the vector $\overrightarrow{\mathbf{b}}' = (D^{-1}\overrightarrow{\mathbf{b}})$, at a cost bounded by $O(t_{M(w,w,1)}, kp_{M(w,w,1)})$ [since $D^{-1}, U' \in \Gamma_{bbt}(T)$]. After that, we may apply Theorem 5.3.1 (see page 113) to compute $U'^{-1}\overrightarrow{\mathbf{b}}'$ at a cost bounded by

$$O\left((\log k)\log w,\ \frac{kw^2}{(\log k)(\log w)}\right).$$

[Note also that $\Gamma_{bbu}(U')$ is available, since $\Gamma_{bbu}(U') \subset \Gamma_{bbt}(T)$.] Combine the two latter bounds to derive the bound (5.19) on the complexity of computing the vector $T^{-1}\overrightarrow{\mathbf{b}}$. ∎

REMARK **5.4.1 (Solving several block bidiagonal linear systems).** When we solve several (say, $q$) nonsingular block bidiagonal linear systems having the same coefficient matrix $T$, we first compute the set $\Gamma_{bbt}(T)$ at a

cost bounded by (5.17), and, after that, we apply Theorem 5.3.1 to simultaneously compute the solution of these $q$ linear systems at a cost bounded by

$$O\left((\log k)\log w,\ \frac{qkw^2}{(\log k)(\log w)}\right). \tag{5.21}$$

REMARK **5.4.2 (Storage space of preprocessing-bbt).** Let $\mu(X)$ denote the storage space required by an arbitrary entity $X$. Observe that

$$\mu\left(S^{(i)}\right) = 2\mu\left(S^{(i-1)}\right),$$

which immediately implies that

$$\mu(S^{(q)}) > \sum_{i=0}^{l} \mu\left(S^{(i-1)}\right) \text{ for any } l < q;$$

$$\mu(\Gamma_{bbu}(U)) = \sum_{i=0}^{r-1} \mu\left(S^{(i-1)}\right) < \mu(S^{(r)}) < \mu(T).$$

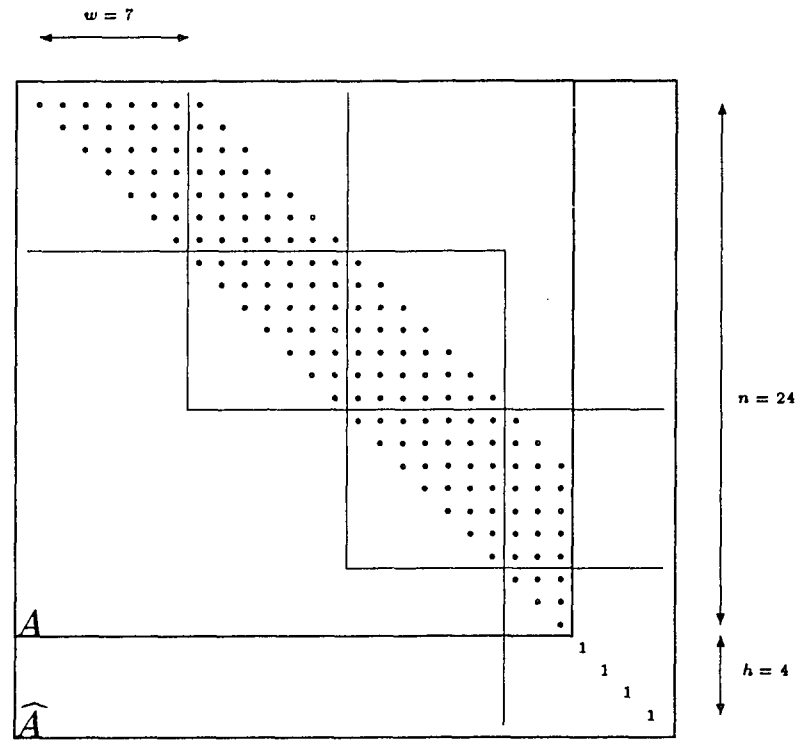
Moreover, $\mu(D) + \mu(U) = \mu(T)$, and, consequently, $\mu(\Gamma_{bbt}(T)) \leq 2\mu(T)$.

## 5.5 Reducing $BAND{\cdot}LIN{\cdot}SOLVE^{\star}$ to solving a block bidiagonal linear system.

In this section, we reduce the problem of solving any nonsingular banded linear system having lower edge to solving a nonsingular block bidiagonal linear system.

We first consider the particular case, where $A$ is an $n \times n$ nonsingular *banded upper triangular* input matrix having the bandwidth $w$. In this case,

Figure 5.1: A $4 \times 4$ block matrix $\hat{A}$ obtained by extending a $24 \times 24$ banded upper triangular matrix $A$ to the desired block bidiagonal format.




if $w$ divides $n$, we may represent the matrix $A$ as an $(n/w) \times (n/w)$ block bidiagonal matrix, with $w \times w$ blocks, where $w = w(A)$. Moreover, we may always shift from the linear system $A\vec{\mathbf{x}} = \vec{\mathbf{b}}$ to the linear system $\hat{A}\vec{\mathbf{y}} = \widehat{\vec{\mathbf{b}}}$ where $\hat{A} = \mathrm{diag}\,(A, I_h)$, with $h$ such that $w$ divides $n + h$, $0 \leq h < w$, and $\widehat{\vec{\mathbf{b}}} = (\vec{\mathbf{b}}, \vec{\mathbf{0}}_h)$, where $\vec{\mathbf{0}}_h$ denotes the $h$-dimensional null vector (see figure 5.1). The bandwidth of the auxiliary matrix satisfies $w(\hat{A}) = w(A) = w$. Once the auxiliary linear system $(\hat{A}\vec{\mathbf{y}} = \widehat{\vec{\mathbf{b}}})$ has been solved,

the solution of the original input linear system is immediately given by the first $n$ components of the solution vector of this auxiliary linear system.

It remains to reduce the problem $BAND \cdot LIN \cdot SOLVE^{\star}$ to the banded triangular case. To this end, denote $p{=}w_{-}(A) \leq w = w(A)$ and define the $(n+p) \times (n+p)$ matrix

$$B = \begin{pmatrix} V & A \\ 0 & W \end{pmatrix}, \tag{5.22}$$

where $V = \begin{pmatrix} I_p \\ 0 \end{pmatrix}$ and $W = (\,0 \quad I_p\,)$. Note that $B$ is a nonsingular triangular matrix having bandwidth $w(B) = w(A)$ (see Figure 5.2, page 120).

We seek a $p$-dimensional vector, denoted $\vec{\mathbf{z}}_0$ and satisfying the following equivalence relation, for any $n$-dimensional vector $\vec{\mathbf{x}}$:

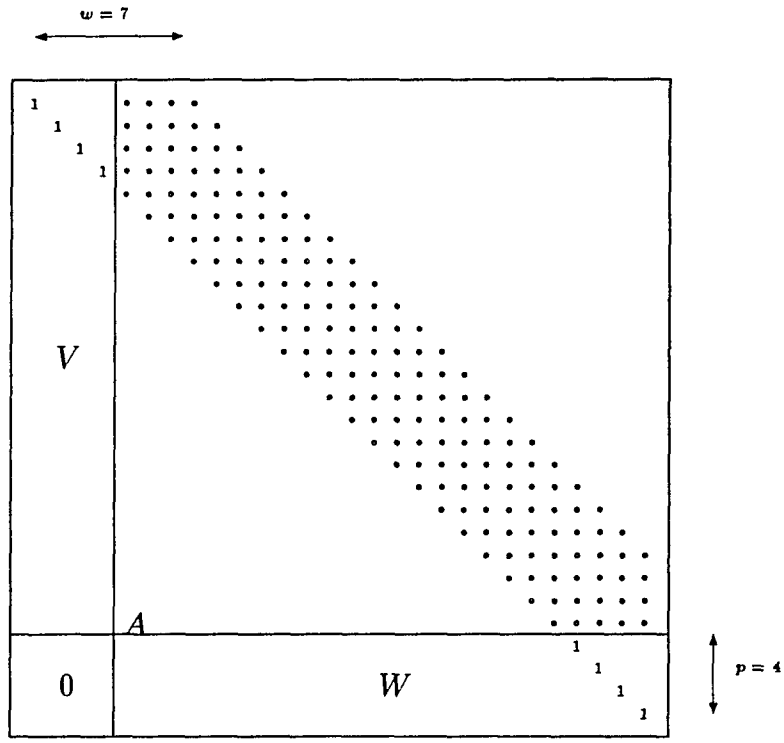
$$\left[A\vec{\mathbf{x}} = \vec{\mathbf{b}}\right] \Longleftrightarrow \left[B\begin{pmatrix} \vec{\mathbf{0}} \\ \vec{\mathbf{x}} \end{pmatrix} = \begin{pmatrix} \vec{\mathbf{b}} \\ \vec{\mathbf{z}}_0 \end{pmatrix}\right]. \tag{5.23}$$

In other words, once such a vector $\vec{\mathbf{z}}_0$ has been computed, the vector $\vec{\mathbf{x}}$ satisfying $A\vec{\mathbf{x}} = \vec{\mathbf{b}}$ can be computed as the solution to the following auxiliary linear system:

$$B\begin{pmatrix} \vec{\mathbf{0}} \\ \vec{\mathbf{x}} \end{pmatrix} = \begin{pmatrix} \vec{\mathbf{b}} \\ \vec{\mathbf{z}}_0 \end{pmatrix}. \tag{5.24}$$

One may solve the latter equation by computing the vector $\vec{\mathbf{y}} = B^{-1}\begin{pmatrix} \vec{\mathbf{b}} \\ \vec{\mathbf{z}}_0 \end{pmatrix}$ (since the vector $\vec{\mathbf{x}}$ is immediately given by the last $n$ components of $\vec{\mathbf{y}}$). The computation of $\vec{\mathbf{y}}$ amounts to solving an upper triangular banded linear system having bandwidth $w$, and, consequently, we may apply to this computation, the algorithm for the banded triangular case discussed at the

Figure 5.2: A $4 \times 4$ block matrix $B$ obtained from a $24 \times 24$ banded matrix $A$, where $w = w(A) = 7$.




beginning of this section. Then, it will remain to compute a vector $\vec{z}_0$ satisfying (5.23). We show in Theorem 5.5.1 (see below) that the required vector $\vec{z}_0$ can be effectively computed, by a deterministic algorithm, at an overall cost bounded by (5.1). Therefore, the linear system $A\vec{x} = \vec{b}$ can be solved at an overall deterministic cost bounded by (5.1) [since the upper triangular linear system (5.24) can be solved at this cost]. ∎

THEOREM **5.5.1** *Let $A$ be a nonsingular $n \times n$ matrix having bandwidth $w$ and having lower edge, $\vec{\mathbf{b}}$ be any given $n$-dimensional vector, and $B, W, V$ be defined by (5.22);*

1. *The relation (5.23) holds for $\vec{\mathbf{z}}_0 = WA^{-1}\vec{\mathbf{b}}$.*

2. *The vector $\vec{\mathbf{z}}_0$ (above) can be computed at a cost bounded by*

$$O\left(\left(\log \frac{n}{w}\right)(\log w) + t_{I(w)}, \frac{\left(\frac{n}{w}\right) t_{I(w)}\, p_{I(w)}}{t_{I(w)} + \left(\log \frac{n}{w}\right)(\log w)}\right). \tag{5.25}$$

PROOF. We immediately deduce part 1 of this theorem by substituting $WA^{-1}\vec{\mathbf{b}}$ for $\vec{\mathbf{z}}_0$ in (5.23). In order to prove part 2, let us denote

$$B^{-1} = \begin{pmatrix} G & H \\ K & L \end{pmatrix}, \tag{5.26}$$

where $H$ is a $p \times p$ matrix. Premultiplying both sides of the equation (5.24) by $B^{-1}$ yields

$$\begin{pmatrix} \vec{\mathbf{0}} \\ \vec{\mathbf{x}} \end{pmatrix} = \begin{pmatrix} G & H \\ K & L \end{pmatrix} \begin{pmatrix} \vec{\mathbf{b}} \\ \vec{\mathbf{z}}_0 \end{pmatrix}. \tag{5.27}$$

Expand the latter equation to derive the vector equation $G\vec{\mathbf{b}} + H\vec{\mathbf{z}}_0 = \vec{\mathbf{0}}$. Due to Proposition 5.5.1 (see below), the matrix $H$ is nonsingular; solving the latter equation for $\vec{\mathbf{z}}_0$ leads to

$$\vec{\mathbf{z}}_0 = -H^{-1}G\vec{\mathbf{b}}. \tag{5.28}$$

It is immediately checked that $-H^{-1}G\vec{\mathbf{b}} = WA^{-1}\vec{\mathbf{b}}$ (see Remark 5.5.1 below). Applying (5.28) to the evaluation of $\vec{\mathbf{z}}_0$, the following computations are sufficient in order to obtain $\vec{\mathbf{z}}_0$:

1. compute the first $w$ rows of $B^{-1}$, at a cost bounded by (5.25), by using Theorem 5.5.2 (see page 124). Note that the matrices $G$ and $H$ are submatrices of the matrix output at this stage (since $p \leq w$);

2. successively compute the matrix $H^{-1}$ and the vector $-H^{-1}G\vec{\mathbf{b}}$, at cost bounded by

$$O\left(t_{I(p)}, p_{I(p)}\right) = O\left(t_{I(w)}, p_{I(w)}\right)$$

and

$$O\left(t_{M(n,p,1)}, p_{M(n,p,1)}\right) = O\left(t_{M(w,w,1)}, \frac{n}{w}p_{M(w,w,1)}\right),$$

respectively [recall that $G$ is a $p \times (n-p)$ matrix and $H$ is a $p \times p$ matrix].

Combine the latter two estimates and the estimate (5.25) on the cost of computing the first $w$ rows of $B^{-1}$ to derive the bound (5.25) on the overall cost of computing the vector $\vec{\mathbf{z}}_0$. It remains to prove that the matrix $H$ is nonsingular, which we will do in the next proposition. ∎

PROPOSITION **5.5.1** *Let $A$ be an $n \times n$ matrix $A$ having bandwidth $w$, having lower bandwidth $w_-(A) = p$ and having lower edge, let $B$ be defined by (5.22), and let $G$, $H$, $K$, $L$ denote the submatrices of $B^{-1}$ defined by (5.26). Then, if $A$ is nonsingular, then $H$ is nonsingular.*

PROOF. The derivation of this proposition is based the following factorizations of the matrices $B$ and $B^{-1}$:

$$B = \begin{pmatrix} I & 0 \\ WA^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & -WA^{-1}V \end{pmatrix} \begin{pmatrix} A^{-1}V & I \\ I & 0 \end{pmatrix}, \qquad (5.29)$$

$$B^{-1} = \begin{pmatrix} 0 & I \\ I & -A^{-1}V \end{pmatrix} \begin{pmatrix} A^{-1} & 0 \\ 0 & -(WA^{-1}V)^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -WA^{-1} & I \end{pmatrix}. \quad (5.30)$$

Expand (5.30) and compare the result of this expansion with (5.26) to deduce that

$$H = -(WA^{-1}V)^{-1}. \quad (5.31)$$

The latter equation immediately implies that $H$ is nonsingular. ■

REMARK **5.5.1** It follows from the equation (5.30) that $G = HWA^{-1}$. To obtain the latter equation, expand (5.30) and compare the blocks of the resulting block matrix with (5.26). Consequently, $\vec{z}_0 = H^{-1}G\vec{b} = H^{-1}HWA^{-1}\vec{b}$ $= WA^{-1}\vec{b}$, which justifies the previous usage of any of expressions $H^{-1}G\vec{b}$ and $WA^{-1}\vec{b}$ for the vector $\vec{z}_0$.

REMARK **5.5.2** All the bounds, established so far for banded linear systems having lower edge, are also valid for banded linear systems having upper edge. The derivation of these bounds, in the case of linear systems with upper edge, are similar to the ones presented for linear systems having lower edge.

REMARK **5.5.3** The equation $H = -(WA^{-1}V)^{-1}$ shows that if a nonsingular matrix $A$ has lower edge, then the south-western $w_-(A) \times w_-(A)$ block of $A^{-1}$ is nonsingular.

REMARK **5.5.4** The vector $\vec{z}_0$ of Theorem 5.5.1 depends on $\vec{b}$ and, therefore, should not be computed at the preprocessing stage. However, the computation of the first $w$ rows of the matrix $B$ should be carried out during

preprocessing, at a cost bounded by (5.1), so that the complexity cost of the subsequent computation of the vector $\overrightarrow{z_0}$, at the backsolving stage, be bounded by (5.2). This two stage computation of the vector $\overrightarrow{z_0}$ enables us to preserve the bounds (5.1) and (5.2) on the cost of solving the problems *PREPROCESS** and *BACK · SOLVE**, respectively.

THEOREM **5.5.2** *If $T$ is a $k \times k$ nonsingular block bidiagonal matrix of the format (5.15), then the complexity of computing $w$ rows/columns of $T^{-1}$ is bounded by (5.17).*

PROOF. The computation of $w$ columns (respectively, rows) of the matrix $T^{-1}$ amounts to solving $w$ linear systems, each having the same coefficient matrix $T$ (respectively, $T^T$); since both $T$ and $T^T$ are block bidiagonal[1], we may apply Theorem (5.4.1) to preprocess the matrix $T$ (respectively, $T^T$) at a cost bounded by (5.17), and, after that, due to Remark 5.4.1, we may compute the required $w$ columns (respectively, rows) at a cost bounded by

$$O\left((\log k)\log w,\ \frac{kw^3}{(\log k)(\log w)}\right)$$

[this bound follows by substituting $w$ for $q$ in equation (5.21)]. ∎

Combining the preprocessing cost bound (5.1), the backsolving cost bound (5.2) and the cost associated with the reduction algorithm of the previous section, we arrive at the following result:

[1]Note that if $T$ is block bidiagonal and block upper triangular, then $T^T$ will be block bidiagonal and block lower triangular.

THEOREM **5.5.3 (Complexity of $BAND \cdot LIN \cdot SOLVE^\star$).** *Any nonsingular $n \times n$ matrix $A$, having bandwidth $w = w(A)$ and having lower and/or upper edge(s), can be preprocessed at a computational cost bounded by (5.1) [by utilizing the reduction algorithm of section 5.5 and by applying Theorem 5.4.1 (see page 114)]. After that, for any fixed $n$-dimension vector $\vec{b}$, the linear system $A\vec{x} = \vec{b}$ can be solved at a computational cost bounded by (5.2).*

## 5.6 Computing the determinant of a banded matrix [with edge(s)].

In this section, we extend the algorithm for the problem $BAND \cdot LIN \cdot SOLVE^\star$ to the computation of the determinant of a banded matrix having lower and/or upper edge(s). Hereafter, $A$ denotes any matrix having bandwidth $w$ and having lower edge (we do not assume that $A$ is nonsingular, as we did in all the previous sections of this chapter); $B$, $V$, $W$ denote the matrices defined by (5.22) (note that the matrix $B$ is nonsingular, due the lower edge assumption); $G$, $H$, $K$, $L$ denote the matrices defined by (5.26).

LEMMA **5.6.1** *If $H$ is nonsingular, then $A$ is nonsingular.*

PROOF. Assume that $H$ is nonsingular. To prove that $A$ is nonsingular, it is sufficient to prove the implication

$$(A\vec{x} = \vec{0}) \Rightarrow (\vec{x} = \vec{0}). \tag{5.32}$$

Assume that $A\vec{x} = \vec{0}$; it follows from the definition of $B$ that

$$B\begin{pmatrix}\vec{0}_p\\ \vec{x}\end{pmatrix} = \begin{pmatrix}V & A\\ 0 & W\end{pmatrix}\begin{pmatrix}\vec{0}_p\\ \vec{x}\end{pmatrix}, \tag{5.33}$$

for any $n$-dimensional vector $\vec{x}$. Expand the right hand side of the above equation to derive the equation

$$B\begin{pmatrix}\vec{0}\\ \vec{x}\end{pmatrix} = \begin{pmatrix}A\vec{x}\\ W\vec{x}\end{pmatrix}; \tag{5.34}$$

recall the assumption ($A\vec{x} = \vec{0}$), and substitute $\vec{0}$ for $A\vec{x}$ in the latter equation to obtain

$$B\begin{pmatrix}\vec{0}\\ \vec{x}\end{pmatrix} = \begin{pmatrix}\vec{0}\\ W\vec{x}\end{pmatrix}; \tag{5.35}$$

premultiplying both sides of the latter equation by $B^{-1} = \begin{pmatrix}G & H\\ K & L\end{pmatrix}$ yields

$$\begin{pmatrix}\vec{0}\\ \vec{x}\end{pmatrix} = \begin{pmatrix}LW\vec{x}\\ LW\vec{x}\end{pmatrix};$$

the latter relation implies that

$$HW\vec{x} = \vec{0}, \tag{5.36}$$

$$\vec{x} = LW\vec{x}. \tag{5.37}$$

Premultiply both sides of the equation (5.36) by $H^{-1}$ to obtain

$$W\vec{x} = \vec{0}; \tag{5.38}$$

combining (5.37) and (5.38), we conclude that $\vec{x} = \vec{0}$. ■

Combining Lemma 5.6.1 (above) and Proposition 5.5.1 (see page 122), we arrive at the following result:

LEMMA **5.6.2** *$H$ is nonsingular if and only if $A$ is nonsingular, or, equivalently, $H$ is singular if and only if $A$ is singular.*

LEMMA **5.6.3**

$$\det A = -(\det B)(\det H). \tag{5.39}$$

PROOF. First observe that, if the matrix $A$ is singular, then, due to Lemma 5.6.2 (above), both sides of the equation (5.39) are obviously zero. It remains to verify that the relation (5.39) also holds in the case where $A$ is nonsingular. Towards this end, apply the factorization (5.29) and the equation (5.31) to deduce that

$$\det B = -(\det A)\,(\det\,(-WA^{-1}V)) = (\det A)/(\det H),$$

which immediately implies that $\det A = -(\det B)(\det H)$. ∎

Let us now estimate the computational complexity of evaluating $\det A$, via formula (5.39):

THEOREM **5.6.1** *The computational complexity of evaluating* $\det A$*, for any $n \times n$ matrix having bandwidth $w$ and having lower edge, is deterministically bounded by (5.4).*

PROOF.

1. compute the matrix $H$, at a cost bounded by (5.17), by using Theorem 5.5.2 (see page 124);

2. once $H$ is available, evaluate $\det H$ at a cost bounded by $\left(t_{D(w)},\ p_{D(w)}\right)$;

3. evaluate

$$\det B = \prod_{i=0}^{n-p} a_{p+i,i}$$

(the product of the entries of $A$ located on the lower edge of $A$) by means of the parallel prefix algorithm, at a cost bounded by $O(\log n, n/\log n)$.

Combine the above bounds to obtain the bound (5.4) on the overall cost of computing $\det A$. ■

# Chapter 6

# Conclusion.

Our original goal ([PSA]), which led to the results presented in this thesis, was to design *parallel algorithms* for solving the problems $BAND \cdot LIN \cdot SOLVE$ and $BAND \cdot DET$, which we wanted to place in $NC^k$ or $RNC^k$ for the smallest constant $k$ and to make them *work optimal* according to the definition of [KR], that is, the *potential work* should be made linear in the complexity of the best known sequential algorithm for the same problem. Even though we fully achieved this goal for the problem $BAND \cdot DET$ (see Table 6.3, page 133), the potential work of our solution to the problem $BAND \cdot LIN \cdot SOLVE$ still exceeds the work required by the best known sequential algorithm by a factor $O(\log n)$, where $n$ is the number of equations in the input linear system. This solution to the problem $BAND \cdot LIN \cdot SOLVE$ substantially improves the previous record bound of [E] [by a factor at least $O(\log^2 n)$]. Moreover, for a large class of instances of the latter problem, we even reached optimality. In particular, this is the case when

the input matrix is over a field the characteristic zero or when it has lower and/or upper edge (see Table 6.1, page 131). However, in the general case, finding an optimal solution to the problem $BAND \cdot LIN \cdot SOLVE$ remains an open problem.

Our main results are summarized in the following tables:

Table 6.1: *BAND · LIN · SOLVE* and *PREPROCESS* (computational complexity estimates).

| Algorithm | \|\| time | Processors | Pot. work | Restrict. |
|---|---|---|---|---|
| Gaussian elim. chap. 1 | $O\left(\frac{n}{w}t_{I(w)}\right)$ | $O\left(p_{I(w)}\right)$ | W | none |
| Prev. rec. (Eberly) chap. 3 | $\Omega\left((\log^3 n)t_{I(n)}\right)$ | $\Omega\left(\left(\frac{n}{w}\right)p_{I(w)}\right)$ | $\Omega\left(\mathrm{W}\log^3 n\right)$ | none |
| $3\times 3$ Solution chap. 3 | $O\left(\left(\log\frac{n}{w}\right)t_{I(w)}\right)$ | $O\left(\left(\frac{n}{w}\right)p_{I(w)}\right)$ | $O(\mathrm{W}\log n)$ | none |
| $2\times 2$ Solution chap. 4 | $O\left(\left(\log\frac{n}{w}\right)t_{I(w)}\right)$ | $O\left(\left(\frac{n}{w}\right)p_{I(w)}\right)$ | $O(\mathrm{W}\log n)$ | none |
| BCR chap. 2 | $O\left(\left(\log\frac{n}{w}\right)t_{I(w)}\right)$ | $O\left(\left(\frac{n}{w}\right)p_{I(w)}/\left(\log\frac{n}{w}\right)\right)$ | W | char(**F**)=0 |
| Reduct. to B2D chap. 5 | $O(T^*(n,w))$ | $O\left(\left(\frac{n}{w}\right)t_{I(w)}\,p_{I(w)}/T^*(n,w)\right)$ | W | edge req. |

$n\times n$ input matrix $A$ over a field $\mathbf{F}$, $w = w(A)$

$\mathrm{W} = O(\frac{n}{w}t_{I(w)}p_{I(w)})$ [best known work bound]

$T^*(n,w) = \left(\log\frac{n}{w}\right)(\log w) + t_{I(w)}$

$\mathrm{char}(\mathbf{F})$ = characteristic of $\mathbf{F}$

Table 6.2: $BACK \cdot SOLVE$, $BACK \cdot SOLVE \cdot SEV$
(computational complexity estimates, after preprocessing).

| Algorithm | \|\| time | Processors | Pot. work | Restrict. |
|---|---|---|---|---|
| Gaussian elim. | $O\left(\frac{n}{w}\log w\right)$ | $O(w^2/\log w)$ | W | none |
| Prev. rec. (Eberly) | | | | |
| $3 \times 3$ Solution | $O\left(\left(\log \frac{n}{w}\right)\log w\right)$ | $O\left(nw/\log w\right)$ | $O\left(\mathrm{W}\log \frac{n}{w}\right)$ | none |
| $2 \times 2$ Solution | $O\left(\left(\log \frac{n}{w}\right)\log w\right)$ | $O\left(nw/\log w\right)$ | $O\left(\mathrm{W}\log \frac{n}{w}\right)$ | none |
| BCR | $O\left(\left(\log \frac{n}{w}\right)\log w\right)$ | $O\left(nw/\left(\left(\log \frac{n}{w}\right)\log w\right)\right)$ | W | char($\mathbf{F}$)=0 |
| Reduct. to B2D | $O\left(\left(\log \frac{n}{w}\right)\log w\right)$ | $O\left(nw/\left(\left(\log \frac{n}{w}\right)\log w\right)\right)$ | W | edge req. |

$n \times n$ input matrix $A$ over a field $\mathbf{F}$, $w = w(A)$
$\mathrm{W} = O(nw)$
char($\mathbf{F}$) = characteristic of $\mathbf{F}$

Table 6.3: $BAND \cdot DET$ and $|BAND \cdot DET|$

(computational complexity estimates).

| Algorithm | $\|$ time | Processors | Pot. work | Restrict. |
|---|---|---|---|---|
| Gaussian elim. | $O(\frac{n}{w}T_D(w))$ | $O(P_D(w))$ | w | none |
| Prev. rec. (Eberly) | $\Omega\left(\left(\log^3 n\right) t_{I(n)}\right)$ | $\Omega\left(\frac{n}{w}P_D(w)\right)$ | $\Omega(\mathrm{w}\log^3 n)$ | none |
| $3 \times 3$ Solution | $O\left(\left(\log \frac{n}{w}\right) T_D(w)\right)$ | $O\left(\frac{n}{w}P_D(w)/\log \frac{n}{w}\right)$ | $O(\mathrm{w})$ | none |
| $2 \times 2$ Solution | $O\left(\left(\log \frac{n}{w}\right) T_D(w)\right)$ | $O\left(\frac{n}{w}P_D(w)/\log \frac{n}{w}\right)$ | $O(\mathrm{w})$ | none |
| BCR | $O\left(\left(\log \frac{n}{w}\right) T_D(w)\right)$ | $O\left(\frac{n}{w}P_D(w)/\log \frac{n}{w}\right)$ | $O(\mathrm{w})$ | char($\mathbf{F}$)=0 $\lvert\det A\rvert$ only |
| Reduct. to B2D | $O\left(T^*(n,w)\right)$ | $O\left(P^*(n,w)\right)$ | $O(\mathrm{w})$ | edge req. |

$n \times n$ input matrix $A$ over a field $\mathbf{F}$, $w = w(A)$

$T_D(w) = t_{I(w)} + t_{D(w)}$, $\quad P_D(w) = p_{I(w)} + p_{D(w)}$

$T^*(n,w) = \left(\log \frac{n}{w}\right)(\log w) + T_D(w)$, $\quad P^*(n,w) = O\left(\frac{n}{w}T_D(w)P_D(w)/T^*(n,w)\right)$

$\mathrm{w} = O\left(\frac{n}{w}T_D(w)P_D(w)\right)$

char($\mathbf{F}$) = characteristic of $\mathbf{F}$

Some of our algorithms (specifically, the ones corresponding to the bounds listed in Tables 6.1 and 6.3, in the rows labeled "$2 \times 2$ solution" and "$3 \times 3$ solution") are not deterministic. Over arbitrary fields, these solutions to the problem $BAND \cdot LIN \cdot SOLVE$ (respectively, $BAND \cdot DET$) utilize randomization of the Las Vegas type (respectively, Monte Carlo type). Over the fields of characteristic zero, these randomized algorithms can be made deterministic. However, in this case [unless the matrix has lower and/or upper edge(s)], it is preferable to utilize the block cyclic reduction algorithm, which deterministically supports an optimum bound, under such a restriction. In the case where the matrix has lower and/or upper edge(s), our algorithm based on reduction to a block bidiagonal linear system provides the fastest work optimum solution and should be used in this case, unless it runs into numerical stability problems.

We will conclude with a brief discussion of some additional open problems related to our results. Throughout our presentation, our characterization of an optimal parallel algorithm relies on the cited definition of [KR]. It would be interesting to consider the optimality of parallel computations with banded matrices by using an alternative definition of the notion of "optimal parallel algorithm", according to which a parallel algorithm is optimal when its work matches a proved lower bound on the sequential complexity of the same problem. Recall (see Tables 6.1 and 6.3) that, for the problems $BAND \cdot DET$ and $BAND \cdot LIN \cdot SOLVE^{\star}$ as well as for the particular case of

$BAND \cdot LIN \cdot SOLVE$, where the input matrix is over a field of characteristic zero, our algorithms are optimal according to the cited definition of [KR], which is a slightly weaker definition of the notion of optimality cited above. These algorithms as well as the block cyclic reduction solution to $BAND \cdot LIN \cdot SOLVE$ also satisfy the stronger definition of optimality, provided that the bandwidth is bounded by a constant, $w = O(1)$. For a general choice of $w$ (say, $w = \Theta(n)$), satisfying the strong definition of optimality would have required to satisfy this definition also for solving a general nonsingular linear system of equations; this well known open research problem is beyond the scope of our work.

Another open problem is the optimality of the parallel solution time. Here again, we reach optimum bound of $O(\log n)$ if $w$ is a constant. For a general choice of $w$ (say, for $w = \Theta(n)$), our algorithms are in $RNC^3$, $NC^3$ or $NC^2$, that is, we are still off by the factors $\log n$ or $\log^2 n$ from the lower bound. This is also partially due to a similar gap for the solution of a general (nonbanded) linear system of $n$ equations.

# Bibliography


[Am] W. F. Ames, *Numerical Methods for Partial Differential Equations*, Academic Press, NY, 1977.

[AHU] V. Aho, E. Hopcroft, D. Ulmam, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass., 1974.

[AR] D. Armon, J. H. Reif, An optimal Space Efficient Parallel Nested Dissection Algorithm, *4th Ann. ACM Symposium on Parallel Algorithms and Architectures*, pp. 344–352, San Diego, CA, 1992.

[AT] M. Augenstein, A. Tenenbaum, *Data Structure and PL/1 Programming*, Prentice Hall, NJ, 1979.

[Ba] S. Barnet, *Matrices Methods and Applications*, Oxford Applied Mathematics and Computing Series, Oxford University Press, NY, 1992.

[BG73] G. Birkhoff, J. George, Elimination by Nested dissection, in *Complexity of Sequential and Parallel Numerical Algorithms*, Traub JB, ed. Academic Press, New York, 1973.

[Br74] R. P. Brent, The Parallel Evaluation of General Arithmetic Expressions, *J. ACM*, 21, 2, pp. 201–206, 1974.

[BGN] B. Buzbee, G. Golub, C. Nielson, On Direct Methods for Solving Poisson's Equations, *SIAM J. Num. Anal.*, 7, 4, pp. 627–655, 1970.

[BD] B. Buzbee, F. Door, The Direct Solution of the Harmonic Equation on Rectangular Regions and the Poisson's Equation on Irregular Regions, *SIAM J. Num. Anal.*, 11, 4, pp. 753–763, 1974.

[BP] D. Bini, V. Y. Pan, *Numerical and Algebraic Computations with Matrices and Polynomials*, Birkhauser, Boston, Mass., 1994 (to appear).

[Co81] S. A. Cook, Towards a Complexity Theory of Synchronous Parallel Computation, *Enseign. Math.*, 27, pp. 99–124, 1981.

[CW] D. Coppersmith, S. Winograd, Matrix Multiplication via Arithmetic Progression, *J. of Symbolic Comp.*, 9, 3, pp. 251–280, 1990.

[DB] G. Dahlquist, Å. Björck, *Numerical Methods*, Prentice-Hall, Englewood Cliff, NJ, 1974.

[E] W. Eberly, On Efficient Band Matrix Arithmetic, *Proc. 33rd Ann. IEEE Symp. on Foundation of Computer Science (FOCS)*, pp. 457–463, 1992.

[EG88] D. Eppstein, Z. Gallil, Parallel Algorithmic Techniques for Combinatorial Computing, *Ann. Rev. Comput. Sci.*, 3, pp. 233–283, 1998.

[Ge73] J. A. George, Nested Dissection of a Regular Finite Mesh, *SIAM J. Numer. Anal.*, 10, pp. 345–367, 1973.

[GeLi] J. A. George, J. W. Liu, *Computer Solution of Large Positive Systems*, Prentice Hall, NJ, 1981.

[GL] G. H. Golub, C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, 1989.

[He] D. Heller, Some Aspects of the Cyclic Reduction Algorithm for Block Tridiagonal Linear Systems, *SIAM J. Num. Anal.*, 13, 4, pp. 484–496, 1976.

[Ho] R. W. Hockney, A Fast Direct Solution of Poisson Equation Using Fourier Analysis, *J. ACM*, 12, pp. 95–113, 1965.

[Kl] G. Klir, *Architecture of Systems Problem Solving*, Plenum Press, New York, NY, 1985.

[KP91] E. Kaltofen, V. Y. Pan, Processor Efficient Parallel Solution of Linear Systems over an Abstract Field, in *Proc. 3rd Ann. ACM Symposium on Parallel Algorithms and Architectures*, pp. 180–191, 1991.

[KP92] E. Kaltofen, V. Y. Pan, Processor Efficient Parallel Solution of Linear System II. The Positive Characteristic and Singular Cases, *Proc. 33rd Ann. IEEE Symp. on Foundation of Computer Science (FOCS)*, pp. 714–723, 1992.

[KR] R. Karp, V. Ramachandran, A Survey of Parallel Algorithms for Shared Memory Machines, in *Handbook for Theoretical Computer Science* (J. van Leeuwen Editor), pp. 869–941, North Holland, Amsterdam, 1990.

[LF] R.E. Ladner, M.J. Fisher, Parallel Prefix Computation, *J. ACM*, 27, 4, pp. 831–838, 1980.

[LPS] J. Laderman, V. Y. Pan, X. H. Sha, On Practical Acceleration of Matrix Multiplication, *Linear Algebra Appl.*, 162–164, pp. 557–558, 1992.

[LRT79] R. J. Lipton, D. Rose, R. E. Tarjan, Generalized Nested Dissection, *SIAM J. Numer. Anal.*, 16, 2, pp. 346–358, 1979.

[LP] L. Lapidus, G.F. Pinder, *Numerical Solutions of Partial Differential Equations in Science and Engineering*, Willey, NY, 1982.

[P] V. Y. Pan, Complexity of Parallel Matrix Computations, *Theoret. Comput. Sci.*, 54, pp. 65–85, 1987.

[PP] V. Y. Pan, F. Preparata, Supereffective Slow-Down of Parallel Computations, *Proc. 4th Ann. ACM Symposium on Parallel Algorithms and Architectures*, pp. 402–409, 1992.

[P91] V. Y. Pan, Complexity of Algorithms for Linear Systems of Equations, in *Computer Algorithms for Solving Linear Algebraic Equations (The State of the Art)*, edited by E. Spedicato, NATO ASI Series, Series F: Computer and Systems Sciences, 77, pp. 27-56, Springer, Berlin, 1991.

[P92a] V. Y. Pan, Complexity of Computations with Matrices and Polynomials, *SIAM Review,* 34, 9, pp. 225-262, 1992.

[P92] V. Y. Pan, Parametrization of Newton Iteration for Computation with Structured Matrices and Applications, *Computers and Mathematics (with Applications)*, 24, 3, pp. 61–75, 1992.

[P93] V. Y. Pan, Parallel Solution of Sparse Linear and Path Systems, in *Synthesis of Parallel Algorithms* (J. H. Reif Editor), pp. 621–678, Morgan Kaufmann publisher, San Mateo, California, 1993.

[PR85] V. Y. Pan, J. H. Reif, Fast and Efficient Parallel Solution of Sparse Linear Systems, *Proc. 17th Ann. ACM Symp. on Theory of Computing*, pp. 402–409, Providence, RI, 1985.

[PR93] V. Y. Pan, J. H. Reif, Fast and Efficient Parallel Solution of Sparse Linear Systems, *SIAM J. Comput.*, 22, 6, pp. 1227–1250, 1993.

[Re93] J. H. Reif, *Synthesis of Parallel Algorithms*, edited by J. H. Reif Editor, Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[PSA] V. Y. Pan, I. Sobze, A. Atinkpahoun, Optimum Parallel Computations with Banded Matrices, *Proc. 5th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pp. 649–658, Arlington, Virginia, 1994.

[Qui] M. J. Quinn, *Parallel Computing*, Theory and Practice, McGraw-Hill, New York, 1994.

[Sc] J.T. Schwartz, Fast Probabilistic Algorithms for Verification of Polynomial Identities, *J. ACM*, 27, 4, pp. 701–717, 1980.

[Sn] M. Snir, On Parallel Searching, *SIAM J. Comput.*, 14, pp. 688–708, 1985.

[St69] V. Strassen, Gaussian Elimination is Not Optimal, *Numer. Math.*, 13, pp. 354–356, 1969.

[Sw] P. Swarztrauber, A Direct Method for the Direct Solution of Separable Elliptic Equations, *SIAM J. Num. Anal.*, 11, 6, pp. 1137–1149, 1974.

[Swe74] R. Sweet, A Generalized Cyclic Reduction Algorithm, *SIAM J. Num. Anal.*, 11, 3, pp. 507–520, 1974.

[Swe77] R. Sweet, A Cyclic Reduction Algorithm for Solving Block Tridiagonal Systems of Arbitrary Dimension, *SIAM J. Num. Anal.*, 14, 3, pp. 706–720, 1977.

[Zi] R.E. Zippel, Probabilistic Algorithms for Sparse Polynomials, *Proc. EUROSAM 79, Lecture Notes in Computer Science, 72*, pp. 216–226, Springer, Berlin, 1979.