Divide and Conquer for the Solution of Banded Linear Systems of Equations

M. Hegland
Computer Sciences Laboratory
Australian National University
Canberra, ACT, 0200

Abstract

An algorithm for the solution of banded linear systems is presented and discussed which combines stability with scalability. This is achieved by implementing divide and conquer for Gaussian elimination with partial pivoting. Earlier divide and conquer algorithms for Gaussian elimination have problems with instabilities and can even break down as they implement a more restricted form of pivoting.

The key observation used for the implementation is the invariance of LU factorization with partial pivoting under permutations. Theoretical analysis shows that the algorithm has low redundancy, a high degree of parallelism and relatively low communication.

1 Introduction

One of the difficulties in the development of parallel numerical algorithms is to maintain the precision of the sequential algorithm while achieving speedup through parallelism. An example of this is the solution of banded linear systems

$$Ax = b \tag{1}$$

with small bandwidths. Whereas sequential methods implement Gaussian elimination with partial pivoting, the suggested parallel methods so far only use restricted forms of pivoting or do not do any pivoting for stability at all [1, 2]. This contrasts with the case of dense linear systems (which are used in the standard LINPACK benchmarks [3]). For dense linear systems partial pivoting is used routinely as the other schemes can lead to poor accuracy and can even break down [4].

Current parallel solvers partition A into blocks. They break down if the diagonal blocks are singular. An example is

$$A = \begin{bmatrix} -1 & 1 & & & \\ 1 & -1 & 1 & & \\ & 1 & -1 & 1 & \\ & & 1 & -1 & \end{bmatrix}$$

which is regular but the diagonal 2x2 blocks

$$A_{1,2} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

are singular.

Near break-down where the diagonal blocks are ill-conditioned while the full matrix is well-conditioned occur frequently in practice. This ill-conditioning of the diagonal blocks has a severe impact on the precision of the current parallel solvers. In the case of

$$A = \begin{bmatrix} q & 1 & & & & \\ 1 & q & 1 & & & \\ & 1 & q & 1 & & \\ & & 1 & q & 1 & \\ & & & 1 & q & 1 \\ & & & & 1 & q \end{bmatrix}$$

with $q = 1.4142$ the rounding error of Gaussian elimination was found to be 10'000 times larger for a typical current parallel method than for the new method suggested here. These situations are very likely to occur in practice for larger matrices. Examples are encountered in the solution of eigenvalue problems by inverse iteration. Thus the method suggested here is certainly competitive as the earlier methods are often incapable of solving the problem.

This deficiency of current parallel banded solvers is addressed in the following and a new approach is suggested which allows the efficient parallel implementation of Gaussian elimination with partial pivoting for banded linear systems. A different approach, although only for the vectorized solution of tridiagonal systems was suggested by Hegland [5]. A disadvantage of the earlier method was the difficulty of the implementation. With the new approach this disadvantage disappears and the implementation compares well with the other divide and conquer methods based on Gaussian elimination.

Basically, the new algorithm has two parts: First the original problem is reformulated. Then a divide

and conquer algorithm is used to solve the reformulated problem. Of course the reformulation has to be such that the precision achievable for the original problem is not compromised. This "stable" reformulation is what distinguishes the new algorithm from earlier attempts.

The computations in the reformulated problem turn out to be exactly the same as they would have been for the original problem but the data locates in different places. In fact, the reformulated problem is obtained from Equation 1 by permutation and is $PAx = Pb$ for an appropriate permutation matrix P. It will be seen that Gaussian elimination with partial pivoting (and a tie breaking strategy) is computationally invariant under permutations. The permutation used here is referred to as reformulation as no data movement is involved, the permutation is purely "virtual".

The reformulated problem is a block bidiagonal system and the diagonal blocks are lower triangular and banded with (lower) bandwidths equal to the sum of the upper and lower bandwidth of the original problem. The number of block-rows of the system is equal to the number of processors and each processor contains one block-row. The divide and conquer method to solve the block bidiagonal system consists of two stages: In the first "concurrent" stage all the independent elimination is done and a block-bidiagonal system with the same number of blocks as the first block-bidiagonal system but with much smaller block sizes is obtained. The second stage consists of a cyclic reduction method to solve the "reduced" block bidiagonal system.

The solution of a linear system of equations usually consists of three stages: First, in the elimination stage the matrix A is factorized, then, using this factorization, the system is solved with forward-elimination and back-substitution [4]. In the following only the elimination stage, i.e., the factorization of the matrix A is discussed. However, the discussion easily extends to the forward-elimination and back-substitution stages as well. The algorithm discussed here also extends to QR factorization, see [6] for a related algorithm for QR factorization on vector processors.

2 Reformulation and shift invariance

This section has two aims: First, to introduce the class of cyclically banded matrices. This class is a superset of the class of banded matrices and has some essential invariance properties under shift permutations which are fundamental to the new algorithms. The second aim of this section is to show that Gaussian elimination with partial pivoting is invariant under shifts.

Definition 1 *A matrix $A = [\alpha_{i,j=1,\ldots,n}]$ is* cyclically banded *with lower bandwidth k_l and upper bandwidth k_u if*

$$\alpha_{i,j} = 0 \quad \textit{if} \begin{cases} \mathrm{mod}(n+j-i,n) > k_u \\ \textit{and} \\ \mathrm{mod}(n+i-j,n) > k_l \end{cases}$$

The set of cyclically banded matrices with lower bandwidth k_l and upper bandwidth k_u is denoted by $\mathcal{B}^n_{k_l,k_u}$ and, for short, $\mathcal{B}^n_k = \mathcal{B}^n_{k,0}$.

Cyclically banded matrices arise, e.g., from discretized boundary value problems with periodic boundary conditions or periodic splines [7]. If $k_u + k_l \geq n$ then $\mathcal{B}^n_{k_l,k_u}$ coincides with $\mathbb{R}^{n,n}$, the set of all $n \times n$ matrices. As an example, the nonzero structure of the matrices in $\mathcal{B}^6_{2,1}$ is:

$$\begin{bmatrix} x & x & & & x & x \\ x & x & x & & & x \\ x & x & x & x & & \\ & x & x & x & x & \\ & & x & x & x & x \\ x & & & x & x & x \end{bmatrix}$$

Banded matrices are examples of cyclically banded matrices. Thus any algorithms for cyclically banded matrices also apply to banded matrices. While there is usually a tradeoff in complexity for the larger generality this is not the case for parallel algorithms (which have the same complexity for banded and cyclically banded matrices).

The bipartite graphs of cyclically banded matrices are rotationally symmetric. This is formulated more concisely using shift matrices. Let S denote the *shift matrix* where $Sx = (\xi_n, \xi_1, \ldots, \xi_{n-1})^T$ for every $x = (\xi_1, \ldots, \xi_n)^T \in \mathbb{R}^n$. Then cyclically banded matrices are invariant under shifts.

Proposition 2 *If $A \in \mathcal{B}^n_{k_l,k_u}$ and S is the shift matrix then*

(a) $SAS^T \in \mathcal{B}^n_{k_l,k_u}$

(b) if $k_u \geq 1$ *then* $SA \in \mathcal{B}^n_{k_l+1,k_u-1}$

(c) $S^{k_u}A \in \mathcal{B}^n_{k_l+k_u}$

Proof: The proof of this proposition is obtained by applying the definitions of S and $\mathcal{B}^n_{k_l,k_u}$. ■

The last statement in Proposition 2 gives rise to the reformulation of the problem. It says that with an appropriate shift any cyclically banded matrix, and, in particular any banded matrix, can be mapped into a cyclically banded matrix with upper bandwidth

$k_u = 0$. In contrast, simple bandedness is not preserved. As a consequence, algorithms for cyclically banded matrices with upper bandwidth 0 can be used to solve systems with cyclically banded matrices with arbitrary bandwidths, including the ordinary banded matrices and this is done by premultiplying the system with a permutation. For example, $\mathcal{B}^6_{1,2}$ is mapped onto $\mathcal{B}^6_{3,0}$ which consists of matrices with the following nonzero structure:

$$\begin{bmatrix} x & & & x & x & x \\ x & x & & & x & x \\ x & x & x & & & x \\ x & x & x & x & & \\ & x & x & x & x & \\ & & x & x & x & x \end{bmatrix}$$

Note that this matrix has no nonzeros in the lower left corner and the only nonzeros in the upper half are in the upper right corner. In the remaining sections parallel algorithms for such matrices will be developed. Before that, it will be demonstrated that permuting the system does not introduce any new numerical errors.

Gaussian elimination [4] is defined as a sequence of elimination steps acting on a sequence of matrices of diminishing size. Each of these steps consists of two stages: In a first stage the pivot row is determined and interchanged with the first row. It is assumed that the choice of the pivot row depends only on the components of the row and not on the row index and is unique. Thus if two potential pivots are of the same size a tie braking strategy is used which depends not on the position of the pivot but only on the matrix entries. The second stage of the elimination process is the actual elimination which mainly consists of the formation of the Schur complement of the pivot. The two stages of a first elimination step are summarized by the factorization

$$\begin{aligned} \hat{P}A &= \begin{bmatrix} \alpha & w^T \\ v & C \end{bmatrix} \qquad (2) \\ &= \begin{bmatrix} 1 & 0 \\ v/\alpha & I_{n-1} \end{bmatrix} \begin{bmatrix} \alpha & w^T \\ 0 & C - vw^T/\alpha \end{bmatrix}. \end{aligned}$$

The factorization then continues by recursively applying this idea until $P_1(C - vw^T/\alpha) = L_1U_1$ is obtained. From this factorization one gets the factorization $PA = LU$ [4] defined by Gaussian elimination with partial pivoting with

$$L = \begin{bmatrix} 1 & 0 \\ P_1^T v/\alpha & L_1 \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} \alpha & w^T \\ 0 & U_1 \end{bmatrix}$$

and $P = \begin{bmatrix} 1 & \\ & P_1 \end{bmatrix} \hat{P}$.

If Gaussian elimination with partial pivoting has the properties just described it is invariant under permutations in the following sense.

Proposition 3 *Let $A' = QA$ for a permutation Q, $PA = LU$ and $P'A' = L'U'$ defined by the LU factorization with partial pivoting. Then $L = L'$, $U = U'$ and $P = P'Q$.*

Proof: The first pivot row has to be the same for A and A' as the rows have only changed in their order by the permutation. Thus the first row of U is the same as the first row of U'. The rest follows by induction. ■

The invariance of Gaussian elimination under permutations can also be used to get distribution-independent algorithms [8].

In view of Propositions 2 and 3 the algorithm to factorize (cyclically) banded matrices $A \in \mathcal{B}^n_{k_l,k_u}$ is

1. Permute A to $B := S^{k_u}A$.
2. Factorize B such that $P_B B = LU$

As a consequence the factorization defined by Gaussian elimination with partial pivoting is obtained as $PA = LU$ with $P = P_B S^{k_u}$. The reformulation step or permutation is done by reinterpreting the meaning of the data in the memory and thus does not require any data movements.

3 Concurrent stage

Divide-and-conquer methods usually have two stages [9]: In the first, concurrent stage independent subproblems are solved without the need for communication. The second stage consists of the combination of solutions of the subproblems to produce the solution to the overall problem. In case of linear equation solvers the subproblems correspond to a representation of the matrix A as a block matrix.

Proposition 4 *Let $n = pq$, $q > k$, $p > 1$ and $A \in \mathcal{B}^n_k$. Then A is block-bidiagonal,*

$$A = \begin{bmatrix} A_1 & & & B_1 \\ B_2 & A_2 & & \\ & \ddots & \ddots & \\ & & B_p & A_p \end{bmatrix} \qquad (3)$$

and the blocks $A_r = \left[\alpha^r_{i,j}\right]$ are banded lower triangular matrices with (lower) bandwidth k, i.e., $\alpha^r_{i,j} = 0$ if $j > i$ or if $i > j + k$, and $B_r = \left[\beta^r_{i,j}\right]$ where $\beta^r_{i,j} = 0$ if $j < q - k + i$.

The blocking is connected with the data distribution and it will be assumed that blocks A_r and B_r will reside on processor r for $r = 1, \ldots, p$.

For example, if $k = 2$ and $q = 6$ then the nonzero structure of the blocks is

$$A_r = \begin{bmatrix} x & & & & & \\ x & x & & & & \\ x & x & x & & & \\ & x & x & x & & \\ & & x & x & x & \\ & & & x & x & x \end{bmatrix}$$

and

$$B_r = \begin{bmatrix} & & & & x & x \\ & & & & & x \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix}.$$

The case $p = 1$ is different as here $A = A_1 + B_1$ where the A_1 and B_1 have the structure defined above.

Now, in order to eliminate the first unknown from the system, a pivot has to be chosen from the first column of A. As the first column of B_2 is zero the pivot must be an entry in A_1, and the subsequent row interchange and even the elimination step does not affect other blocks than A_1 and B_1 and, in particular, can be done on Processor 1 without any consultations with the other processors.

With the same reasoning, the $q+1$st unknown can be eliminated independently on Processor 2 and the same holds for the $2q+1$st unknown on Processor 3 etc. Furthermore, all these elimination steps can be done concurrently. After one concurrent elimination step eliminating one unknown on each processor the remaining Schur complement is of order $n-p = p(q-1)$. If $q = k+1$ the concurrent part of the algorithm is then completed. In the case $q > k+1$ the concurrent step has to be repeated $q-k-1$ times. After conclusion of the concurrent part, the cyclic reduction part, which is described in the next section, can start.

The matrix factorization which summarizes the operations done on processor r during the concurrent elimination of the unknowns $1, q+1, 2q+1, \ldots$ is

$$\begin{aligned} \hat{P}_r A_r &= \begin{bmatrix} \alpha_r & w_r^T \\ v_r & C_r \end{bmatrix} \qquad (4) \\ &= \begin{bmatrix} 1 & 0 \\ v_r/\alpha_r & I_{q-1} \end{bmatrix} \begin{bmatrix} \alpha_r & w_r^T \\ 0 & C_r - v_r w_r^T/\alpha_r \end{bmatrix} \end{aligned}$$

and

$$\begin{aligned} \hat{P}_r B_r &= \begin{bmatrix} u_r^T \\ D_r \end{bmatrix} \qquad (5) \\ &= \begin{bmatrix} 1 & 0 \\ v_r/\alpha_r & I_{q-1} \end{bmatrix} \begin{bmatrix} u_r^T \\ D_r - v_r u_r^T/\alpha_r \end{bmatrix}. \end{aligned}$$

The Schur complement does not have the original nonzero structure anymore. A little wave can appear in A_r and the nonzero block of B_r becomes rectangular. The following defines this structure more concisely and it is suggested to call this a wave matrix. Wave matrices are invariant under concurrent Schur complements.

Definition 5 *A matrix A is a* wave matrix *with bandwidth k and p waves if it is block bidiagonal,*

$$A = \begin{bmatrix} A_1 & & & B_1 \\ B_2 & A_2 & & \\ & \ddots & \ddots & \\ & & B_p & A_p \end{bmatrix}$$

and if $A_r = [\alpha_{ij}]$, $B_r = \left[\beta_{ij}^r\right]$ and $\alpha_{ij}^r = 0$ if $i > j+k$ or $j > \max(i,k)$ and $\beta_{ij}^r = 0$ if $k < \max(i, q-j)$. The set of wave matrices is denoted by $\mathcal{W}_k^{n,p}$.

For example, if $n = pq$ and $q = 6$, $k = 2$ then the diagonal blocks A_r have the structure

$$\begin{bmatrix} x & x & & & & \\ x & x & & & & \\ x & x & x & & & \\ & x & x & x & & \\ & & x & x & x & \\ & & & x & x & x \end{bmatrix}$$

and the offdiagonal blocks B_r have the structure

$$\begin{bmatrix} & & & & x & x \\ & & & & x & x \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix}.$$

Proposition 6 *Let $A \in \mathcal{W}_k^{n,p}$ be a wave matrix and let*

$$A' = \begin{bmatrix} A_1' & & & B_1' \\ B_2' & A_2' & & \\ & \ddots & \ddots & \\ & & B_p' & A_p' \end{bmatrix}$$

be the reduced system after one concurrent elimination step, i.e.,

$$A_r' = C_r - v_r w_r^T/\alpha_r, \quad B_r' = D_r - v_r u_r^T/\alpha_r$$

where C_r, D_r, u_r, v_r, w_r and α_r are defined in Equations (4) and (5). Then A' is a wave matrix, in particular $A' \in \mathcal{W}_k^{n-p,p}$.

The number of floating point operations does depend on the choice of the pivot. Two cases can occur: Either the pivot resides in rows 1 to k. In this case the row interchange does not change the nonzero structure of A_r or B_r. In this case the computation of the Schur complement requires $2k(k-1)$ floating point operations for A'_r, $2(k-1)k+k$ floating point operations for B'_r plus k divisions for v_r/α_r. This gives a total of $2k(2k-1)$ floating point operations for the first case.

In the second case the pivot row is the row $k+1$. After an interchange with the first row the nonzero structure is modified. In this case the forming of A'_r requires $2k(k-1)+k$ floating point operations, B'_r requires 0 floating point operations and again k divisions are required for v_r/α_r. This gives a total of $2k^2$ floating point operations.

Note that actually the "worst case" where the row $k+1$ is chosen as pivot requires less floating point operations than the others! For an estimate the maximum has to be taken which is $2k(2k-1)$. In total, each processor has to do $q-k$ steps such that the number of floating point operations required per processor is

$$fl/p = (q-k)2k(2k-1)$$

or summed over all p processors

$$fl = p(q-k)2k(2k-1) = (n-pk)2k(2k-1).$$

This floating point operation count is an upper bound and with careful coding it could be decreased substantially for special cases, for example, the matrix B_r can slowly disappear during the elimination process if the original matrix is large enough and diagonally dominant enough (see [5] for a method which takes this into account for tridiagonal matrices). Diagonal dominance has also been used to generate early terminating cyclic reduction, see [10]. A new parallel solver using diagonal dominance is the overlapped partitions method (OPM) [11, 12].

The effects of early termination or incomplete factorization still remain to be analyzed in detail for the method suggested here. In addition, overlapped partitioning might lead to very fast and stable method if combined with the algorithm presented here. Another open question is to how the "wave" is represented and how the case of no wave is treated. It is suggested that sparse matrix methods could be applied.

4 Cyclic reduction

After completing the concurrent steps one is left with a block bidiagonal matrix A':

$$A' = \begin{bmatrix} A'_1 & & & B'_1 \\ B'_2 & A'_2 & & \\ & \ddots & \ddots & \\ & & B'_p & A'_p \end{bmatrix}$$

Processor i contains one diagonal block A'_i and the corresponding offdiagonal block B'_i in its local memory. All the blocks are dense k by k matrices. This system is then solved by a variant of cyclic reduction. Assume for simplicity that the number of processors is a power of 2. Then $\log_2(p)$ reduction steps are required to complete the factorization.

Each reduction step has two phases: A communication phase and a computation phase. In the *communication phase* half of the processors send their blocks to the other half of the processors. More precisely, processor $2i-1$ either sends its blocks A_{2i-1} and B_{2i-1} to the corresponding even numbered processor $2i$ or receives the blocks A_{2i}, B_{2i} from processor $2i$ where $i = 1, \ldots, p/2$. Choosing sender and receiver can be done such as to increase the amount of local communication and decrease non-local communication in hierarchical networks.

Mathematically, the communication phase corresponds to a re-blocking of the matrix:

$$A' = \begin{bmatrix} \tilde{A}'_1 & & & \tilde{B}'_1 \\ \tilde{B}'_2 & \tilde{A}'_2 & & \\ & \ddots & \ddots & \\ & & \tilde{B}'_{p/2} & \tilde{A}'_{p/2} \end{bmatrix}$$

where

$$\tilde{A}'_i = \begin{bmatrix} A'_{2i-1} & 0 \\ B'_{2i} & A'_{2i} \end{bmatrix}$$

and

$$\tilde{B}'_i = \begin{bmatrix} 0 & B'_{2i-1} \\ 0 & 0 \end{bmatrix}, \quad i = 1, \ldots, p/2.$$

In the *computational stage* a concurrent elimination of the first k unknowns corresponding to each $\tilde{A}_i$ can be done. This stage has some similarity with the earlier concurrent elimination stage, however, the structure of the blocks is slightly different.

After the concurrent elimination stage another communication stage of the next reduction step redistributes the matrix to $p/4$ processors and a concurrent elimination again halves the number of unknowns and

so on until only one processor has a dense system of order $2k$ which is then factorized. A characteristic feature of this type of reduction algorithm is that the degree of parallelism [9] is halved at every step.

At each step $2k^2$ numbers are communicated between sender and receiver. The number of floating point operations is

$$fl = \sum_{j=1}^{k} 2(2k-j)^2 + (2k-j) = \frac{14}{3}k^3 - \frac{3}{2}k^2 - \frac{1}{6}k.$$

At the end a $2k$ by $2k$ system has to be solved. This requires

$$fl = \sum_{j=1}^{2k-1} 2j^2 + j = \frac{16}{3}k^3 - 2k^2 - \frac{1}{3}k$$

floating point operations.

5 Scalability

The scalability of parallel algorithms is limited by various factors including redundancy, degree of parallelism and communication overhead. These factors shall now be discussed for the new algorithm.

The number of floating point operations is clearly dependent on the algorithm. Sequential algorithms usually have been optimized to minimize operation counts. These optimized algorithms often do not parallelize well and other algorithms with more floating point operations have to be chosen for parallel execution. This is also the case for the algorithm studied here which is not optimal with respect to the floating point operation count. The ratio of floating point operations required for the algorithm used for p processors and the one used on 1 processor is called *redundancy* [9]. For Gaussian elimination with pivoting the floating point operation count can depend heavily on the numerical values. In the following, the floating point operation count is understood to be the best upper bound on floating point operations which can be obtained if nothing about the numerical values of the nonzero matrix entries is known.

By summing up the floating point operation counts obtained in the previous sections on gets that the total floating point operation count of the new algorithm for $A \in \mathcal{W}_k^{n,p}$ on p processors:

$$fl_p = 2k(2k-1)n + \frac{k}{6}(4k-1)(k+1)p - k^2(4k-1).$$

In a similar way one gets the floating point operation count for $A \in \mathcal{W}_k^{n,1}$ on one processor:

$$fl_1 = 2k(2k-1)n - \frac{k}{3}(4k-1)(2k-1).$$

The redundancy is then approximately in the case of very small bandwidths, i.e., $k^3 \ll n$:

$$r_{p,n,k} = fl_p/fl_1 = 1 + \frac{k(4k-1)(k+1)}{12k(2k-1)}\,\frac{p}{n}.$$

In particular, $r_{p,n,2} = 1 + 0.58\frac{p}{n}$ and $r_{p,n,4} = 1 + 0.9\frac{p}{n}$. Asymptotically in n there is thus no redundancy for fixed p and small k. Thus the algorithm is very scalable from this point of view.

The picture is different if the algorithm is to be used to solve ordinary banded linear systems. For a banded linear system with lower bandwidth k_l and upper bandwidth k_u the floating point operation count is [1] approximately for small k_u and k_l:

$$fl_1 = (2k_l + 2k_u + 1)k_l\, n.$$

The approximate floating point operation count for the new parallel algorithm is obtained by substituting $k = k_u + k_l$:

$$\begin{aligned} fl_p &= 2(k_l+k_u)(2k_l+2k_u-1)\, n \\ &+ \frac{k_l+k_u}{6}(4k_l+4k_u-1)(k_l+k_u+1)\, p. \end{aligned}$$

In this case the redundancy is for small bandwidths:

$$\begin{aligned} r_{p,n,k_u,k_l} &= \frac{2(k_l+k_u)(2k_l+2k_u-1)}{(2k_l+2k_u+1)k_l} \\ &+ \frac{(k_l+k_u)(4k_l+4k_u-1)(k_l+k_u+1)}{6(2k_l+2k_u+1)k_l}\,\frac{p}{n}. \end{aligned}$$

Asymptotically for large n one gets

$$r_{p,\infty,k_u,k_l} = \frac{2(k_l+k_u)(2k_l+2k_u-1)}{(2k_l+2k_u+1)k_l}.$$

This is bounded from above by $2 + 2\frac{k_u}{k_l}$ so for banded matrices with $k_u = k_l$ the upper bound of the redundancy is 4. For tridiagonal and pentadiagonal matrices one has $r_{p,\infty,1,1} = 2.4$ and $r_{p,\infty,2,2} = 3.1$. These asymptotic ratios compare well to the redundancies for Gaussian elimination without pivoting or Cholesky factorization which can be found in [1, 13].

Another determining factor for the scalability of an algorithm is the *average degree of parallelism*. It is seen to be [9]

$$a_{p,n,k} = \frac{fl_p}{\sum_{j=1}^{p} \frac{fl_p^{(j)}}{j}}$$

where fl_p is the total amount of floating point operations and $fl_p^{(j)}$ is the amount of floating point operations which are done concurrently on j processors.

For the algorithm discussed one gets from the previous sections

$$fl_p^{(j)} = \begin{cases} \frac{16}{3}k^3 - 2k^2 - \frac{1}{3}k, & \text{for } j = 1 \\ \frac{14}{3}k^3 - \frac{3}{2}k^2 - \frac{1}{6}k, & \text{for } j = 2, 4, \ldots, \frac{p}{2} \\ 2k(2k-1)(n/p-k) & \text{for } j = p \\ 0 & \text{else.} \end{cases}$$

If n/p is large most of the time is spent concurrently on all p processors. The average degree of parallelism is for small bandwidth and large numbers of processors p:

$$a_{p,n,k} = \left(\frac{1 + \frac{(4k-1)(k+1)}{12(2k-1)} \frac{p}{n}}{1 + \left(\frac{28k^2-9k-1}{12(2k-1)} \log_2(p) - 3k/2 \right) \frac{p}{n}} \right) p.$$

This is for tridiagonal systems:

$$a_{p,n,2} = \frac{1 + 0.58p/n}{1 + (2.6 \log_2(p) - 3)p/n} p.$$

This shows that the algorithm is well balanced for large problem sizes, e.g., $a_{10^3,10^6,2} = 980$. For pentadiagonal systems $k = 4$ and so

$$a_{p,n,4} = \frac{1 + 0.9p/n}{1 + (5 \log_2(p) - 6)p/n} p,$$

for example, $a_{10^3,10^6,2} = 950$.

Even an algorithm which has low redundancy and a high average degree of parallelism can still perform very poorly on a distributed memory computer if it requires a high amount of *communication*. This is quite often the real bottleneck of many algorithms including the one discussed here. The reason for this is that communication speeds are often orders of magnitudes lower than computation speeds.

Communication is needed in the cyclic reduction phase of the elimination algorithm. This is done in $\log_2(p) - 1$ steps where $2k^2$ floating point numbers are communicated per sender. A simple model for the time required to send m floating point numbers from any processor to any other is

$$t_{\text{comm}} = \alpha + \beta m$$

where α is the latency and β is the communication throughput. With this the ratio of communication time to computation time is modelled as

$$\kappa = \frac{6(\log_2(p) - 1)\frac{\alpha}{k\rho} + 12k(\log_2(p) - 1)\frac{\beta}{\rho}}{6(2k-1)(2\frac{n}{p} - 3k) + \log_2(p)(28k^2 - 9k - 1)}$$

It is seen that for large enough n this can be made arbitrarily small but for practical cases this might be extremely large. As the data is sent in relatively small packets the communication latency is usually the determining part, especially for very small bandwidths.

6 Conclusions

By reformulating banded linear systems as cyclically banded linear systems with zero upper bandwidth a parallel algorithm is developed which includes partial pivoting. Redundancy is no higher than for comparable algorithms which do not use pivoting and the algorithm shows good scalability properties. The redundancy is partly due to "fill-in" in the nonzero structure of the matrix and further investigations will show how incomplete factorizations can reduce this fill-in and with that reduce redundancy. At the moment this algorithm is implemented on the Fujitsu AP 1000 [14]. The numerical computations are identical to the computations done to do an LU factorization with partial pivoting of a wrapped-around matrix but the data storage and movement has been chosen such that the algorithm is easily implemented. The algorithm can be described as a multi-frontal algorithm [15] for the parallel solution of banded linear systems.

Acknowledgements

The author would like to thank M. Osborne, ANU, who suggested to study block-bidiagonal systems for the solution of banded linear systems. Furthermore, P. Arbenz, ETH Zürich, and A. Cleary, University of Tennessee, assisted by carefully reading the manuscript and suggesting several improvements. The research was partially supported by the "Advanced Computational Systems" CRC.

References

[1] P. Arbenz and W. Gander, "A survey of direct parallel algorithms for banded linear systems", Tech. Rep. 221, Departement Informatik, ETH Zürich, 1994.

[2] J.M. Ortega, *Introduction to Parallel and Vector Solution of Linear Systems*, Plenum Press, New York, 1988.

[3] J.J. Dongarra, "Performance of various computers using standard linear equation software", Tech. Rep., CS Dept., University of Tennessee, 1992.

[4] G.H. Golub and C.F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, 2nd edition, 1989.

[5] M. Hegland, "On the parallel solution of tridiagonal systems by wrap-around partitioning and incomplete LU factorization", *Numerische Mathematik*, vol. 59, no. 5, pp. 453–472, 1991.

[6] D. Dun, M. Hegland, and M. Osborne, "Parallel stable solution methods for banded linear systems", Tech. Rep., Centre Math. Appl. and Comp. Sci. Lab., Australian National University, 1995.

[7] C. de Boor, *A Practical Guide to Splines*, Springer, 1978.

[8] M. Hegland, "A distribution independent algorithm for the reduction to tridiagonal form using one-sided rotations", in *Proceedings of the IEEE First International Conference on Algorithms And Architectures for Parallel Processing*, 1995, vol. 1, pp. 286–289.

[9] K. Hwang, *Advanced Computer Architecture*, McGraw Hill, 1993.

[10] D. Heller, "Some aspects of the cyclic reduction algorithm for block tridiagonal linear systems", *SIAM J. Numer. Anal.*, vol. 13, no. 4, pp. 484–496, 1976.

[11] J.-L. Larriba-Pey, A. Jorba, and J.J. Navarro, "OPM: a parallel method to solve banded systems of equations", Tech. Rep. UPC-CEPBA-92-09, Universitat Politècnica de Catalunya, 1992.

[12] J.-L. Larriba-Pey, A. Jorba, and J.J. Navarro, "Solution of strictly diagonal dominant tridiagonal on vector computers", Tech. Rep. UPC-DAC-93-18, Universitat Politècnica de Catalunya, 1993.

[13] A.J. Cleary, "Parallelism and fill-in in the Cholesky factorization of reordered banded matrices", Tech. Rep. SAND-90-2727, Sandia National Laboratories, Albuquerque, New Mexico, 1990.

[14] H. Ishihata, T. Horie, S. Inano, T. Shimizu, and S. Kato, "Cap-ii architecture", in *First Fujitsu-ANU CAP Workshop*, 1990.

[15] I.S. Duff, A.M. Erisman, and J.K. Reid, *Direct Methods for Sparse Matrices*, Clarendon Press Oxford, 1986.