PARALLEL COMPUTING 83
M. Feilmeier, J. Joubert and U. Schendel (eds.)
Elsevier Science Publishers B.V. (North-Holland)

SYSTOLIC MATRIX ITERATIVE ALGORITHMS

M.Berzins, T.F. Buckley and P.M. Dew

Department of Computer Studies
The University
Leeds LS2 9JT
U.K.

Systolic algorithms for matrix iterative methods are described for systems of linear equations that often arise in the discretisation of partial differential equations. In particular the Gauss-Seidel iterative method with a relaxation factor is considered in detail for an irregular sparse-banded coefficient matrix. The systolic algorithms compute m iterations in $\Theta(2n)$ time units using $m \times b$ processing cells, where b is the number of nonzero bands in the coefficient matrix.

INTRODUCTION

It is well known that the numerical solution of nonlinear partial differential equations involves the solution of large sparse systems of linear equations that often arise as one step of a modified Newton's method. It follows therefore that an important first step in the design of fast parallel algorithms/architectures for partial differential equations is the development of parallel algorithms for the solution of sparse systems of linear equations. In this paper we consider the design of *systolic algorithms* for matrix iterative methods, where the coefficient matrix is an *irregular sparse-banded* matrix (see below). The main reason for considering iterative methods is that it is much easier to illustrate how to exploit the sparsity in the matrix. We are concerned here with the design of the high level systolic algorithms, for details on some of the implementation issues the reader is referred, in the first instance, to Dew (1983). The original motivation for this work arose out of a study of VLSI architectures for the simulation of gas transmission networks supported by the British Gas Corporation (see Dew, Buckley and Berzins (1983)).

The general problem we wish to consider is the solution of a large sparse system of banded equations written in the standard form

$$A\underline{x} = \underline{b}$$

where the $n \times n$ coefficient matrix A is assumed to be *diagonally dominant* so that it is well-posed for a matrix iterative algorithm. The type of coefficient matrix that commonly occurs from the finite difference discretisation of partial differential equations in simple regions (e.g. Laplace's equation) has a series of non-zero bands. Such matrices are referred to as *sparse-banded* matrices. Systolic iterative algorithms for this type of matrix was first considered by Kung and Leiserson (1980) and more recently by Dew, Buckley and Berzins (1983). In this paper we generalise the algorithms to handle matrices where the bands do not necessary run diagonally across the matrix, but the overall bandwidth is small compared with the order of the matrix n. Matrices of this type often arise from the discretisation of partial differential equations, providing that the nodes are ordered in a suitable manner.

Definition

A *nxn* square matrix A is said to be *sparse-banded*, if there exist a sequence of bands running diagonally across the matrix such that for $j \in N_n = [1, 2, \cdots, n]$

$$[D_0]_j = a_{j,j}$$

$$[U_i]_j = a_{j,j+m_i(j)} \quad i = 1, 2, \cdots, k$$

$$[L_i]_j = a_{j,j-m_{-i}(j)} \quad i = 1, 2, \cdots, l$$

where $a_{i,j}=0$ whenever $i,j \notin N_n$, and there exist positive constants p_i and q_i such that for all j

$$m_i(j)=p_i, \quad m_{-i}(j)=q_i$$

and the sequences $\{p_i\}_1^k$ and $\{q_i\}_1^l$ are strictly monotonically increasing with $p_k=p$ and $q_l=q$ so that the total bandwidth of the matrix is given by $b_w=p+q+1$. The sparseness comes from the fact that k and l are assumed small compared with n.

The matrix is said to be an *irregular sparse-banded*, if $m_i : N_n \rightarrow N_n$ is one-one and the sequences, $\{m_i(j)\}_{i=1}^k$, $\{m_{-i}(j)\}_{i=1}^l$ are strictly monotonically increasing for each $j \in N_n$, and in addition satisfy the inequalities, for all j

$$m_i(j) \leq p_i \text{ and } p_i - m_i(j) \leq d$$

$$m_{-i}(j) \geq q_i \text{ and } m_i(j) - q_i \leq d$$

where d is a positive constant. □

An example of an irregular sparse-banded matrix arising in the simulation of gas transmission networks can be found in Dew, Buckley and Berzins (1983). The systolic approach used in this paper is particularly well suited to the case when $b_w \ll n$.

SYSTOLIC MATRIX ITERATIVE ARRAY

We shall consider the *Jacobi method* given by

$$D\underline{x}^{(\tau+1)} = -(L + U)\underline{x}^{(\tau)} + \underline{b} \qquad \tau = 0, 1, 2, \cdots$$

and the *Gauss-Seidel method* with relaxation factor ω,

$$(D + \omega L)\underline{x}^{(\tau+1)} = ((1-\omega)D - \omega U)\underline{x}^{(\tau)} + \underline{b} \quad \tau = 0, 1, 2, \cdots$$

where L, D, U denote the usual lower triangular, diagonal and upper triangular matrices of A and $\underline{x}^{(0)}$ is the initial vector.

To derive and describe the systolic arrays we shall apply the programming technique of *stepwise refinement.* The first level is simply to note that we can perform m iterations systolically by simply using a linear systolic array to perform each iteration (which is essentially a matrix/vector multiplication). This is shown in figure 1.

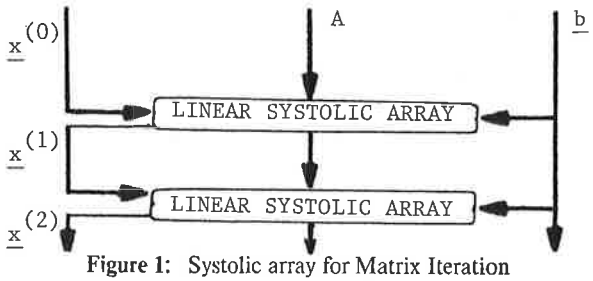

Figure 1: Systolic array for Matrix Iteration

The reader should notice that the above array satisfies the systolic property that the I/O is only from the boundaries of the array. The next stage of the refinement is the design of the linear systolic array to perform one iteration. For this we need systolic arrays to perform the computations

$$S_u(\underline{c},\tau) = U\underline{x}^{(\tau)} + \underline{c} \text{ and } S_l(\underline{c},\tau) = L\underline{x}^{(\tau)} + \underline{c}$$

and a boundary cell to combine them with the appropriate diagonal element. For the Jacobi iterative array we pipe the result of $S_l(\underline{b},\tau)$ to give $S_u(S_l(\underline{b},\tau),\tau)$ and then divide by the diagonal element (see figure 2a). In the case of the Gauss-Seidel method we have to compute $S_u(\underline{b},\tau)$ and $S_l(\underline{0},\tau+1)$ separately and use a boundary cell to perform the rather more complex operation

$$\underline{x}^{(\tau+1)} = (1-\omega)\,\underline{x}^{(\tau)} - \omega\, a_{j,j}^{-1}\{S_u(\underline{b},\tau) + S_l(\underline{0},\tau+1)\}$$

This is shown in figure 2b. The systolic arrays for S_u and S_l can be interleaved in the manner used by Kung (1980) for the IIR filter computation but this can be regarded as a lower level refinement.

IRREGULAR SPARSE MATRIX/VECTOR MULTIPLICATION

So far we have defined the main data paths through the array. In the next level of refinement we add the timing information. Following the work of Leiserson and Saxe (1983) we shall use directed graphs to describe

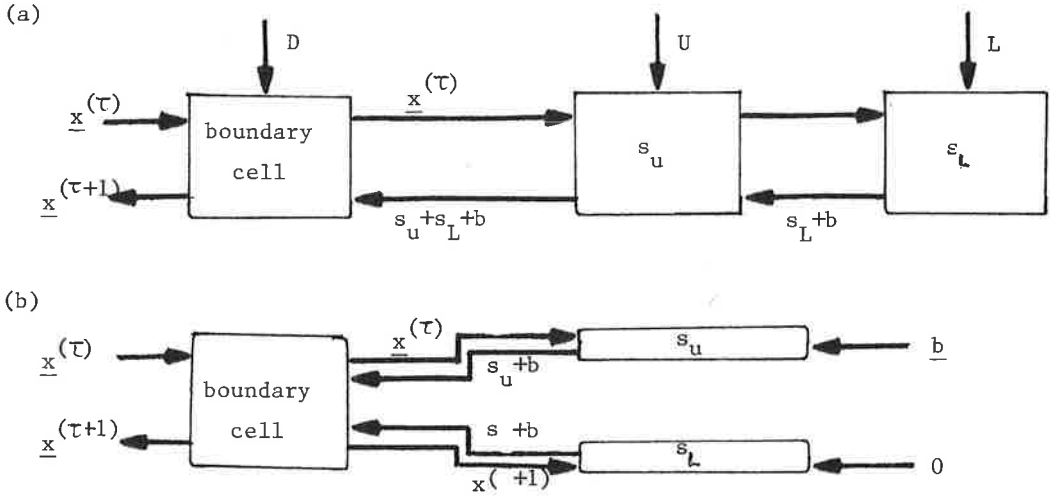

Figure 2: Systolic Arrays for Performing One Iteration

the systolic algorithms, where the weights on the edges represent delays (i.e. registers) and the nodes represent each computational step or I/O port. The directed graph for the upper triangular matrix/vector

multiplication, $S_u(\underline{b}, \tau)$, is shown in figure 3. The circular nodes denote one inner product step computation and the square nodes are the I/O ports. The data is "twiced-slowed" which means that the results arrive every other time cycle. and a null element is inserted after each element in the input data streams in the standard systolic manner. To minimise the pipeline latency we would set $c = m_k(j) - (k-1)/2$ where $x_{j+1/2}$ denotes the null element after x_j assuming j is an integer value. Applying the node retiming technique given in Leiserson and Saxe (1983) it is easy to verify that the network will compute the required results.

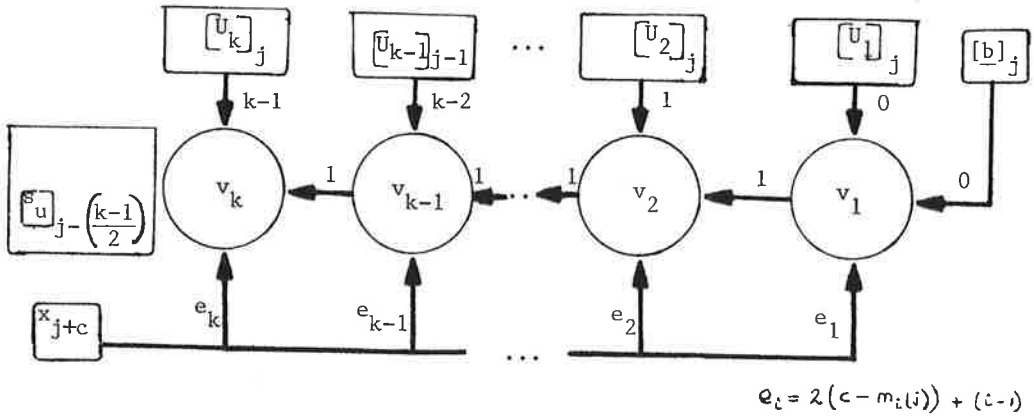

Figure 3: The Network for Computing $S_u(\underline{b}, \tau)$

An immediate consequent of having an *irregular sparse-banded* matrix is that the delays are dependent on the timing parameter j. This means that we most now provide *positional* information for each coefficient of the matrix. For example each upper diagonal element of the matrix, $a_{j,j+m_i(j)}$, would be represented by a data structure that contains its numerical value and the distance $d_i = p_i - m_i(j)$. In a *semisystolic algorithm* we would maintain a window of active x-values and use the positional information to *route* the appropriate elements of $\underline{x}^{(\tau)}$ to the inner product step cells. Alternatively we can adopt a truly systolic approach and maintain an active window within each cell. This is achieved by retiming the array for a sparse-banded matrix with constants, p_i, and saving in each cell the last d x-values (ignoring the null elements). The positional information can then be used to select the required value of $\underline{x}^{(\tau)}$. In the definition of a irregular sparse-banded matrix we have been careful to ensure that d_i is bounded above by d. The systolic array for a sparse-banded upper matrix/vector multiplication is shown in figure 4, remembering that the input/output data streams are "twiced slowed".

We can adopt a similar design procedure for the lower triangular matrix/vector multiplication. For the Gauss-Seidel iterative method we must be careful to design the array so that the pipeline latency is zero or more precisely at the end of the period when x_j arrives we must output $[S_l(\underline{0}, \tau + 1)]_{j+1}$. Again we can handle the irregularity in the sparsity pattern by either using a semisystolic design or by transforming the array to handle a sparse-banded matrix with constants q_i. This time we save upto the last d matrix coefficients arriving at each cell. This systolic array is shown figure 5.

SYSTOLIC ARRAY FOR GAUSS-SEIDEL ITERATION

The systolic arrays for computing $S_u(\underline{b}, \tau)$ and $S_l(\underline{0}, \tau+1)$ can now be combined with a boundary cell which performs the computation given by equation 2. This is shown in figure 6. The boundary cell also outputs the updated vector to the array below and to the array computing S_l. Notice that the use of the directed graph and the sub-division of the problem makes it easier to determine the delays and verify the correctness of the design. For example, to compute the delay for the elements of A and $\underline{b}$ between iteration rows we note that if the input to the array is $x_{j+c+1}^{(\tau)}$ then the input to the next array is $x_{j-(k+1)/2}^{(\tau+1)}$ where $c=p-(k-1)/2$. The required delay is given by

$$p' = 2(j+c+1) - 2(j-(k+1)/2) = 2(p+1)$$

The pipeline latency for an array performing m iterations is given by $mp'-1$. A similar type of array can be constructed for the Jacobi iterative method ,see Dew (1983).

The systolic algorithm shown in figure 4 is now in a form where we can address the implementation issues. For example, pipelining in the inner-product cells and the in balance between the computation the boundary

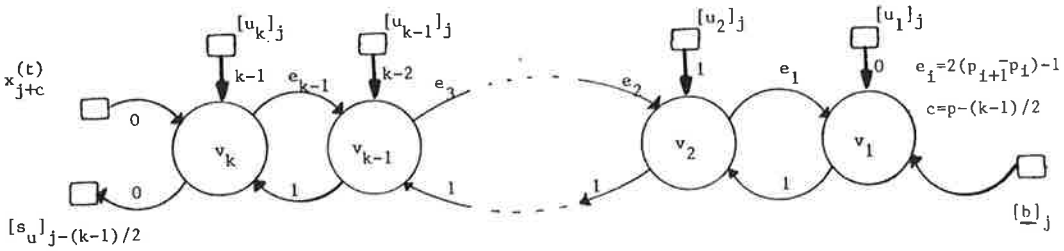

Figure 4: Systolic Array for Sparse-Banded Upper Triangular Matrix/Vector Multiplication

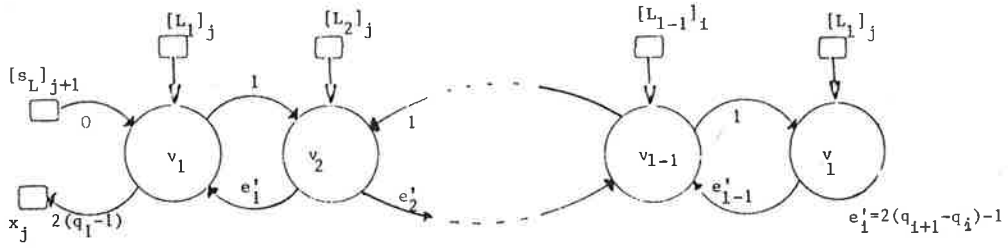

Figure 5: Systolic Array for Sparse-Banded Lower Triangular Matrix/Vector Multiplication

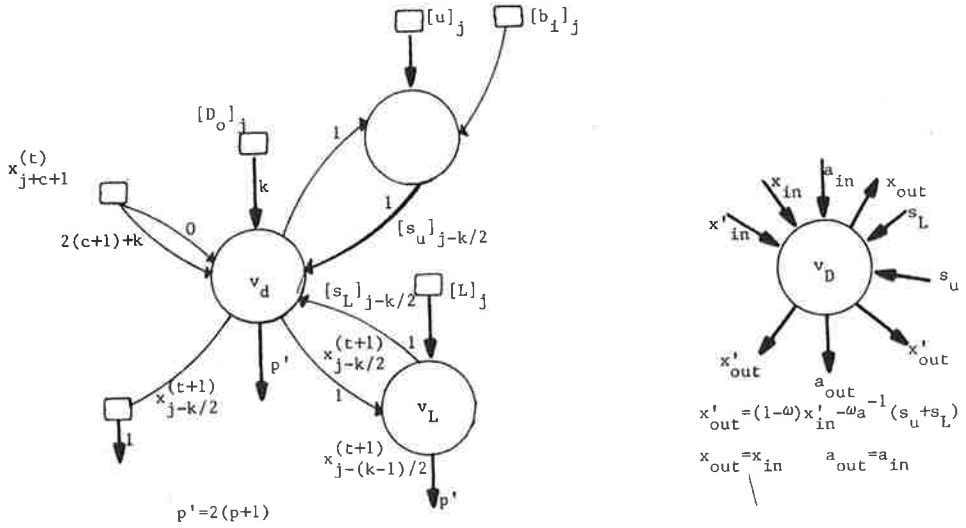

Figure 6: Systolic Array for Gauss-Seidel Method

cell performs and inner product step cells. This is beyond the scope of this paper but these problems can normally be handled by slowly down the data flow rate through the array. This is particularly effective when we have multiple right hand sides, $AX = B$, since the inner pipelines can be kept full by inputing the elements of B row by row. This has the added advantage that the data flow rate of the matrix coefficients can be r times slower than the rate for the $\underline{x}^{(\tau)}$ and $\underline{b}$, where r is the number of columns in the matrix B.

ACKNOWLEGDEMENT

The authors would like to thank the British Gas Corporation for supporting Martin Berzins during the period this research was carried out.

REFERENCES

[1] Dew, P.M., Buckley, T.F. and Berzins, M., Application of VLSI devices to computational problems in the gas industry, Tech. Report 163, Dept. of Comp. Studies, Leeds Univ. (1983).

[2] Dew, P.M., VLSI architectures for problems in numerical computation, in: Paddon D.J. and Pryce, J.D. (eds), Workshop on progress in the use of vector and array processors (Academic Press, to appear).

[3] Kung, H.T. and Leiserson C.E., Systolic arrays (for VLSI), in: Duff, I.S. and Stewart, G.W. (eds), Sparse Matrix Proceedings 1978, (1979) (A sightly different version in Introduction to VLSI systems by C.A. Mead and L.A. Conway, Addison-Wesley 1980)

[4] Kung H.T., Special-purpose devices for signal and image processing, in: Real-Time Signal Processing III, (Society of Photo-Optical Instrumentation Engineers, 1980)

[5] Leiserson, C.E. and Saxe, J.B., Optimizing synchronous systems, Jrnl of VLSI and Computer Systems, 1, (1983), 41-67.