SYSTOLIC ALGORITHMS FOR SOLVING LINEAR SYSTEMS

Chau-Jy Lin, Muh-Cherng Wu and Chih-Han Lin
Department of Applied Mathematics,
Department of industrial engineering and management,
National Chiao-Tung University,
Hsinchu, 30050, Taiwan, Republic of China
E-mail: cjlin@math.nctu.edu.tw

Abstract
Let $A, B$ be two arbitrary $n \times n, n \times m$ matrices. We present a parallel algorithm to solve the dense linear system $AX=B$. The computational model used is a systolic array consisting of $n(n+1)$ processing elements (PEs). The total number of PEs used is independent of $m$, the number of columns in matrix $B$. The pivot equation can be changed during the execution of our systolic algorithm. When $A$ is nonsingular, the total time step of our algorithm is $4n+m-1$. If $A$ is singular, a simple triangular matrix will be used to detect whether there is no solution or many solutions. The elapsed time within a time step is independent of $n$ and $m$.

1 Introduction

In the past decade, it has witnessed a tremendous explosion of research on various aspects of parallel computers. Thus, many problems are solved on parallel computers by parallel algorithms, such as the arithmetic evaluation problems, the combinatorial problems and the matrix operation problems [3,4,9,11]. One of the parallel computers is the systolic array. Systolic arrays are represented by graphs with nodes and edges. The nodes correspond to the processing elements (PEs) and the edges to the data communication links between PEs. An algorithm can be executed on a systolic array is called as a systolic algorithm. Many systolic algorithms have been presented to solve various problems [2,7,10,12].

Let $A, B$ be two arbitrary $n \times n, n \times m$ matrices, respectively. We present a systolic algorithm to obtain the $m$ solutions of the dense linear system $AX=B$. The solution of a linear system is important for many scientific and engineering problems. Many articles had discussed how to solve this linear system [1,5,6,8,9]. Usually, a linear solver requires the assumption such that $A$ is symmetric positive definite. In our algorithm, if $A$ is nonsingular, the output data are the desired $m$ solutions. However, if $A$ is singular, the output will form a simple linear system. It can be used to determine whether there is no solution or many solutions for each column of $B$. Since this simple linear system produces an upper triangular matrix, it can be solved easily by the way of backward substitution. This solving way has the same behavior in our systolic algorithm. Under the use of Gauss-Jordan elimination method to solve $AX=B$, we have to choose an equation as a pivot equation (PIE). This PIE is used to eliminate the coefficients of a specified unknown variable within another equations. If a PIE is used to eliminate the coefficients of the $i$th unknown $x_i$ of other equations, the coefficient of $x_i$ in this PIE is called as the pivot header (PIH). If a PIH is zero, then this PIE is degenerated into a null pivot equation (NPIE).

2 THE COMPUTATIONAL MODEL

There are total $n(n+1)$ PEs to be used in our computational model. It is arranged as a two
dimensional array. Each PE is referred to as PE(i,j) for 1 ≤ i ≤ n, 1 ≤ j ≤ n+1. See Figure 1. Thus, our computational model can also be considered as n rows of linear systolic array. Each row of linear array has (n+1) PEs. There are exactly two data communication links, say a-link and p-link, between the PEs in our computational model. The a-link connects PE(i,j) to PE(i+1,j-1). The p-link connects PE(i,j) to PE(i,j+1). The purposes of these two links are listed as follows.

(1) The a-link carries the modified equation of AX=B from the i-th linear systolic array to the (i+1)th linear systolic array. In this case, the variable x_i is deleted.

(2) The p-link carries a PIE on the i-th linear systolic array.

At the beginning of our algorithm, the coefficients of the j-th equation of AX=B is arranged to appear at the input of a-link of PE(1,j), for 1 ≤ j ≤ n. Thus, the number of PEs is independent of m. If we distribute the coefficients of an equation to the PEs of the first row, then the number of PEs in this row will be n+m.

In our systolic array, a time step is defined as an enough large time unit such that any PE can perform the following three tasks.

(1) PE(i,j) reads the input data from PE(i-1,j+1) or PE(i,j-1) via a-link or p-link, respectively.

(2) PE(i,j) executes our designed algorithm exactly once loop.

(3) PE(i,j) sends the output data to PE(i+1,j-1) or PE(i,j+1) via a-link or p-link, respectively.

During the execution of our systolic algorithm, if PE(i,j) sends its output data at a time step t, then these output data will become the input data of PE(i+1,j-1) or PE(i,j+1) at the time step t+1.

3 THE STRUCTURES OF PES

The PEs used in our computational model are classified into four types. In the i-th linear array, for 1 ≤ i ≤ n, (1) the PE(i,1) is in type-1; (2) the PE(i,n+1) is in type-2; (3) for (n-i+2) ≤ j ≤ n, the PE(i,j) is in type-3; (4) for 2 ≤ j ≤ n-i+1, the PE(i,j) is in type-4.

Thus, the i-th linear systolic array consists of one PE of type-1, one PE of type-2, (i-1) PEs of type-3 and (n-i) PEs of type-4.

Let a_in and a_out denote the input and output data on the a-link, respectively. Similarly, the above definitions can be applied to p_in and p_out. Each PE contains a register H to store the first real number which appears at the a_in (or a_out) of the type-2, 3, 4 (or type-1). The PE of type-4 needs another register F to indicate whether a new PIE appears or not. Once a new PIE appears, the register F is also used to indicate whether the old PIE is an NPIE or not. The structures of PEs are indicated in Figure 2, where type-4(H,F) means that these PEs contain the registers H and F.

The major tasks of PEs within the i-th linear array are listed as follows.

(1) In the type-1, the equation on the a-link of PE(i,1) is default set as a PIE. That is, PE(i,1) sends the data on a-link to its p-link in order to eliminate the coefficients of the unknown variable x_i. This PIE is an NPIE when the first a_in of a-link is zero.

(2) In the type-2, PE(i,n+1) only transforms the data from its p-link to a-link.

(3) In the type-3, using the PIE on the p-link, PE(i,j) eliminates a coefficient
of the unknown variable $x_i$.

(4) In the type-4, PE$(i,j)$ tries to produce a new PIE. If it is false, then PE$(i,j)$ performs the same work as in the type-3. If the try is affirmative, then PE$(i,j)$ performs the work as in the type-1 and type-3.

4 THE DESIGN OF ALGORITHM

For $1 \leq i \leq n$, the $i$th linear systolic array is used to eliminate the coefficients of $x_i$. When $A$ is nonsingular, the coefficient of $x_i$ of the last PIE is modified into the value of 1. However, this value of 1 is not sent into the $(i+1)$th linear systolic array. Therefore, the $i$th linear systolic array produces $(n-1)$ equations without the term $x_i$ and one equation with an implicit term $x_i$.

At the initial state of our algorithm, a PE waits for a real datum to be appeared at its $p_{in}$ (or $a_{in}$) for the type-2, 3, 4 (or type-1). The first time step of a PE to meet a real datum of $p_{in}$ (or $a_{in}$) is called as the time step of beginning-actual-evaluation (BAE) in this PE. We use the time step $t_b$ to denote the BAE of a PE. The time step before the BAE is in the waiting state. We use a symbol "*" in the $p$-link (or $a$-link) to mean the waiting state. The major operations of PEs are considered as two cases. One is at the time step $t=t_b$. The other is at the time step of $t>t_b$.

Case (I): at the time step $t=t_b$.

1. All PEs store $p_{in}$ (or $a_{in}$) into its register H for the type-2, 3, 4 (or type-1)
2. The PE in the type-2, 4 sends the "*" to its $a_{out}$. That is, between two consecutive linear systolic arrays, we use a waiting symbol "*" to denote the deleted coefficient of a unknown variable.
3. In the PE of type-3, if its $p_{in}$ is not zero, the PE does the same task as that of (2). However, if the PE has its $p_{in}$ = 0, then the $a_{in}$ will be sent to its $a_{out}$. In this case, $A$ is singular.
4. The PE of type-1 sends its $a_{in}$ to $p_{out}$.
5. In the type-3 or the type-4 without a new PIE, the PE sends its $p_{in}$ to $p_{out}$. Here, we use the condition of $|a_{in}|>|p_{in}|$ as our strategy to obtain a new PIE.
6. In the of type-4 with a new PIE, the PE sends its $a_{in}$ to $p_{out}$. Also, this PE sets a indication to the register F. If $p_{in}$=0, then we set F=1. If $p_{in}$=0, then we set F=2. Note that $p_{in}$ = 0 means that the old PIE is an NPIE.

Case (II): at the time step $t>t_b$.

7. In the type-1, if $H$=0, that is, an NPIE has appeared, then the PE only sends $a_{in}$ to $p_{out}$. Otherwise, PE sends $a_{in}/H$ to its $p_{out}$.
8. In the type-2, the PE just sends $p_{in}$ to $a_{out}$.
9. In the type-3 or the type-4 without a new PIE, the PE sends $p_{in}$ to $p_{out}$. And this PE modifies its $a_{in}$ by $p_{in}$ to obtain its $a_{out}$.
10. In the type-4 with a new PIE, the PE sends the value of $a_{in}/H$ to $p_{out}$. And this PE modifies $p_{in}$ by $a_{in}$ to obtain its $a_{out}$. The action of modification depends on the value of F. If $F$=2, then PE only sends $p_{in}$ to $a_{out}$. If $F$=1 then PE sends the result of $p_{in}$-$a_{in}/H$ to $a_{out}$.
11. All PEs do the above processes of (6)-(9) until a stopping signal "^" appears at $p_{in}$ (or $a_{in}$) for the type-2, 3, 4 (or type-1).

5. THE SYSTOLIC ALGORITHM

Some procedures are defined as follows.
\begin{verbatim}
procedure waiting-data-on-links \equiv a_{in}=a_{in};
p_{in}=-^; break.
procedure stopping-signal-on-links \equiv a_{out}=-^;
p_{out}=-^; break.
procedure normal-modifying-equation \equiv p_{out}=p_{in}; a_{out}=a_{in}-H*p_{in}.
procedure null-pivot-or-not-3 \equiv p_{out}=p_{in}; if (p_{in}=0) then \{ H=0; a_{out}=a_{in}; \}
else \{ H= a_{in}; a_{out}=-^; \}
procedure no-new-pivot-equation-4 \equiv p_{out}=p_{in}; H= a_{in}; a_{out}=-^;
procedure changing-pivot-equation-4 \equiv H= a_{in}; p_{out}=a_{in}; a_{out}=-^.
if (p_{in}=0) then F=2 else F=1.
procedure other-modifying-equation \equiv a_{out}=p_{in}-a_{in}/H; p_{out}=a_{in}/H.

ALGORITHM Linear-solver-for-AX=B \equiv

[Initial state]
For all PEs, we have H=-^; Let F=0 in the type-4.
For 1 \leq j \leq n, the PE(1,j) has (j-1) waiting time steps on the input of \( a \)-link. That is, the PE(1,j) has
\( a_{in}=-^ \) at the time step \( t \) for 1 \leq t \leq j-1. The \( j \)th equation of AX=B is arranged as the \( a_{in} \) of PE(1,j)
from the time step \( t=j \) to the time step \( t=j+n+m-1 \).
At the time step \( t=n+m+1 \), PE(1,1) has \( a_{in}=-^ \) in order to stop the execution of PE in our algorithm.

[executive state]
repeat \( f^* \) do parallel for all PEs. \( f^*/
if (a_{in}=-^ \cup p_{in}=-^) then waiting-data-on-links;
if (a_{out}=-^ \cup p_{out}=-^) then stopping-signal-on-links;
switch (type of PE)
{
  case “1”:
    if (H=-^) then \{ H= a_{in}; p_{out}=a_{in}; \}
  else \{ if (H \neq 0) then p_{out}=a_{in}/H else p_{out}=a_{in}; \}
    break;
  case “2”:
    if (H=^) then \{ H=p_{in}; a_{out}=^ \} else a_{out}=p_{in};
    break;
  case “3”:
    if (H=^) then null-pivot-or-not-3 else normal-modifying-equation;
    break;
  case “4”:
    if (H=^) then
      if (|a_{in}| = |p_{in}|) then changing-pivot-equation-4
      else no-new-pivot-equation-4;
    if (H \neq 0) then
      (if (F=0) then normal-modifying-equation ;
      if (F=1) then other-modifying-equation;
      if (F=2) then \{ a_{out}=p_{in}; p_{out}={a_{in}}/{H}; \}
      )
    until (a_{in}=-^ \cup p_{in}=-^).

Some observations in our algorithms are listed as follows.

(1) At the initial time, we set H=-^ in any PE. At the
initial time \( t=t_b \), the first real number \( p_{in} \) (or \( a_{in} \)) arrives at a PE. We set H=\( p_{in} \) (or H=\( a_{in} \)).
Thus, we use \( H \neq 0 \) to indicate the time step \( t > t_b \) within the switch statement of our algorithm.

(2) At the initial state, in the PEs of type-4, we have F=0. When a PE has its F \neq 0, it means
that the input of its \( a \)-link carries a new PIE. Thus, the action of modifying equation is
different from that of PEs in the type-3. If F=0 is kept, then the PE of the type-4
performs the same procedure of normal-modifying-equation as that of the
PEs in the type-3.

(3) In the procedure other-modifying-equation,
\end{verbatim}
we had obtained a new PIE. This implies that the content of \( H \) is not zero by the condition of \( |a_{in}|>|p_n| \) to obtain a new PIE. Thus, the value of \( a_{in}/H \) is always valid.

4) If the first \( a_{in} \) of \( PE(i,1) \) is zero, then its \( p \)-link carries an NPIE. This NPIE will be replaced by a new PIE unless all the followed \( (n-i) \) equations are absent of the term \( x_i \). In this case, we will obtain \( H=0 \) in \( PE(i,n+1) \). The \( H=0 \) in \( PE(i,n+1) \) indicates that the matrix \( A \) is singular. Once \( A \) is singular, we require more effort to determine whether there is no solution or infinitive solution for each column of \( B \). In this case, from the \( a_{out} \) of the \( n \)th linear array, we obtain a new linear system with a simple upper triangular matrix.

For a fixed \( i, 1 \leq i \leq n \), during the execution of our algorithm, we obtain the following results.

1) The \( PE(i,1) \) has its \( t_b=3(i-1)+1 \).

2) For \( 1 \leq j \leq n+1 \), the \( PE(i,j) \) has its \( t_b=3(i-1)+j \).

3) In the \( t \)th linear array, if an NPIE exists, then it appears just once time.

4) If \( PE(i,1) \) has an NPIE and all type-4 \( PE(i,j) \) have \( a_{in}=0 \) at the time steps \( t=3(i-1)+j \), then \( PE(i,n+1) \) has its \( H=0 \).

5) For \( 1 \leq j \leq n \), if \( A \) is nonsingular, \( PE(i,j) \) has \( (n+m-i+1) \) real data on its input of \( a \)-link.

6) For \( 2 \leq j \leq n+1 \), if \( A \) is nonsingular, \( PE(i,j) \) has \( (n+m-i) \) real data on its output of \( a \)-link.

7) The \( j \)th linear array eliminates \( (n-1) \) terms of \( x_i \) within \( (n-1) \) equations except in the last PIE.

8) When the content of \( H \) is not zero, the \( PE(i,n+1) \) sends an equation to \( PE(i+1,n) \) with an implicit term \( x_i \).

9) The equation with the implicit term \( x_i \) will be carried to \( PE(n,i+1) \).

10) When \( A \) is nonsingular, the \( a_{out} \) of \( PE(n,j+1) \) is the solution of \( x_j \) in the linear system \( AX=B \).

11) The \( PE(i,1) \) has its stop signal “^” at \( t=n+m+2(i-1)+1 \).

12) For \( 1 \leq j \leq n+1 \), the \( PE(i,j) \) has its stop signal “^” at \( t=n+m+2(i-1)+j \).

13) Our algorithm requires the total time step of \( t=4n+m-1 \), when \( A \) is nonsingular.

14) If \( A \) is nonsingular, the execution of our algorithm \( Linear-solver-for-AX=B \) is correct to obtain the desired \( m \) solutions.

6 DISCUSSION

When \( A \) is singular, some \( PE(i,n+1) \) has its \( H=0 \).

Thus, some data of \( a \)-link in the PEs of type-3 can not be deleted. From the procedures \( null-pivot-or-not-3 \) and \( waiting-data-on-links \), these undeleted data will be sent out from the \( n \)th linear array. This implies that the output will be formed a new linear system of \( n \) equations.

Now we consider the \( j \)th equation of the new linear system. From our algorithm, if \( H=0 \) in \( PE(i,n+1) \) and \( j \neq i \), then the \( j \)th equation can not be deleted its variable \( x_j \). Hence the \( j \)th equation of new system has to contain a term of \( x_j \). Under this consideration, the equations of the new linear system can be divided into two parts according to the condition that if the \( H \) of \( PE(i,n+1) \) is zero or not. Let \( k \) be the numbers of \( H \) such that \( H=0 \) in \( PE(i,n+1) \) with \( i=I(1), I(2), ..., I(k) \). We assume that \( I(1)<I(2)<...<I(k) \). The first part system contains these \( k \) equations with indexes \( I(1), I(2), ..., I(k) \). It has the following form, where the coefficients \( c_{ij} \) are issued by the procedure \( null-pivot-or-not-3 \) and are carried out by the procedure \( waiting-data-on-links \).

\[
0x_{b(1)}+c_{12}x_{b(2)}+c_{13}x_{b(3)}+...+c_{1b_k}x_{b(k)}=b_{b(1)} \\
0x_{b(2)}+c_{23}x_{b(3)}+...+c_{2k}x_{b(k)}=b_{b(2)} \\
...
\]

223
The remaining \((n-k)\) equations form the second part system. For an equation with index \(j\) in the second part system, let \(I(r)\) be the smallest index in the first part system such that \(j < I(r)\). We have the equation with the form of

\[ x_j + c_{j,k(r+1)}x_{k(r+1)} + c_{j,k(r+2)}x_{k(r+2)} + \ldots + c_{j,k}x_{k} = b_j. \]

Therefore, if the linear system (***) has been solved, then \(x_j\) will be obtained from the method of substitution.

Now we solve the system (***) in the same way. If the value of \(b_{k}^j\) is not zero, then there is no solution. If \(b_{k}^j=0\), then we mark that the variable \(x_{k}^j\) is arbitrary. The linear system (***) becomes a system with \((k-1)\) equations. Its corresponding matrix is an upper triangular matrix. Let it be denoted as the matrix \(S\).

The corresponding solution will be performed by the method of backward substitution. In fact, this backward substitution will be designed as the same as our systolic algorithm without the PEs of type-4. When \(S\) is nonsingular, the system \(AX=B\) has many solutions with the variable \(x_{k(1)}\) being arbitrary. If \(S\) is also singular, then a recursive algorithm on \(S\) is continuous until we obtain the final result.

7 CONCLUSIONS

We present a systolic algorithm to solve the dense linear system \(AX=B\) for \(A\) and \(B\) being two arbitrary \(n \times n\) and \(m \times n\) matrices, respectively. The computational model used is a systolic array consisting of \(n(n+1)\) PEs. The total number of PEs used is independent of \(m\). The pivot equation can be changed during the execution of our systolic algorithm. When \(A\) is nonsingular, the total time step of our algorithm is \(4n+m-1\). If \(A\) is singular, the method of backward substitution with the similar systolic algorithm will detect whether there is no solution or many solutions. The elapsed time within a time step is independent of \(n\) and \(m\). Every PE has a simple structure and the PE of a type executes the same instructions within any time step. We hope that the design consideration and the result of our algorithms can be applied to solve other problems, such as the problems within the fields of numerical method, the domain decomposition technique, the eigen-value system, the computational geometry, the differential equation, the graph theory and so on.

Acknowledgments: This work was partially supported by National Science Council, Taiwan, Republic of China, under the contract number: NSC-89-2115-M-009-012.

References


Figure 1. The systolic array for linear system.

Figure 2. The structures of PEs.