STUDIA UNIV. BABEŞ-BOLYAI, MATHEMATICA, XXXVII, 3, 1992

# PARALLEL PRE-PROCESSING IN EXTRAPOLATION METHODS FOR SOLVING ORDINARY DIFFERENTIAL EQUATIONS

O. BRUDARU[*] and G.M. MEGSON[**]

Received: December 1, 1992
AMS Subject Classification: 68Q22, 58F99

**Abstract:** - The work deals with the parallel implementation of the pre-processing stage of the polynomial/rational extrapolation techniques for solving initial value problems in ordinary differential equations. The multiple use of the Euler method is analysed in the case of three step number sequences $(n_k)$: $n_k=ak+b$, $n_k=2**k$, and $n_{k+1}=n_k+n_{k-1}$, $k=0,..,K$. We make use of the PRAM model of parallel computation and propose a specific parallel algorithm for each type of sequence. It is assumed that the number of processors is a factor of $K$. The performances concerning the parallel processing time and the effectiveness of processor utilization are established. Some conditions ensuring the optimality of the proposed algorithms are also given.

**Keywords:** ordinary differential equations, polynomial and rational extrapolation, parallel algorithms.

## 1. Introduction.

The parallel implementation of the polynomial and rational extrapolation techniques ([10], [12-13]) for solving initial value problems in ordinary differential equations (ODE's) is discussed in [8] where two types of systolic arrays are proposed.

We consider here a complementary task, namely, the design of parallel algorithms for computing the initial data input to the systolic arrays in [8].

The stages involved by the extrapolation methods for solving initial value problems in ODE's are described in section 2. In section 3, we present three parallel algorithms to compute the data entering the extrapolating process, and consider Euler's

[*] Universitatea "Al. I. Cuza", Seminarul Matematic "A. Myller" 6600-Iaşi, Romania

[**] Computing Laboratory, University of Newcastle-Upon-Tyne Claremont Tower, Claremont Rd, Newcastle-Upon-Tyne, NE1 7RU, U.K.

method applied to three types of step number sequences. The performances concerning the parallel processing time and the effectiveness of processor utilization are established. Some comments are given in the last section.

**2. Extrapolation methods for solving ODE's.** Consider the initial value problem

$$y'=f(t,y),\quad t_0\le t\le b,\quad y(t_0)=y_0; \tag{2.1}$$

and let us suppose that we need to compute $y(t_0+H)$, $t_0+H\le b$. Let $y(t)$ be the true solution of (2.1) and $y(t,h)$ be the approximate solution obtained by using step length $h$, $h\le H$, and a suitable numerical method which is applied many times. Let us take some step-number sequence $n_0<n_1<n_2<\dots$, put $h_k=H/n_k$ and define

$$Y(k,0)=y(t_0+H,h_k), \tag{2.2}$$

the numerical solution obtained by performing $n_k$ steps with step size $h_k$, $k=0,2,\dots,K$. The computation of the $Y(k,0)$-values represents the pre-processing stage of the extrapolation.

Let us suppose that $y(t,h)$ admits an asymptotic expansion in $h$ of the following form

$$y(t,h)=y(t)+d_1h^{r(1)}+d_2h^{r(2)}+\dots+d_mh^{r(m)}+\dots \tag{2.3}$$

where $0<r(1)<r(2)<\dots<r(m)$, $d_1,d_2,\dots$ are independent of $h$.

**2.1.** Polynomial and rational extrapolation. If we consider $r(i)=ir$ in (2.3) then following [13] the polynomial extrapolation involves the computing of the $Y$-values table given by

$$Y(k,m)=Y(k+1,m-1)+[Y(k+1,m-1)-Y(k,m-1)]/[(n_{k+m}/n_k)^r-1] \tag{2.4}$$

for $k=0,1,\dots,K-m$ and $m=1,2,\dots,K$. The computation given by (2.4) is represented (for $K=5$ and $m=5$) in Figure 1.

For $n_k=H/(h_0b^k)$, $b\epsilon(0,1)$, (2.4) becomes

$$Y(k,m)=Y(k+1,m-1)+[Y(k+1,m-1)-Y(k,m-1)]/[1/b^{mr}-1], \qquad (2.5)$$

$k=0,1...,K-m$ $m=1,2,...,K$.

Usually, $r=1$ or $r=2$. The process is stopped ([13]) when

$$\text{abs}(Y(0,m)-Y(0,m-1))\leq\text{tol}, \qquad (2.6)$$

where tol is a prescrbied tolerance and $Y(0,m)$ approximates $y(t_0+H)$.

The rational extrapolation ([10]) is defined by

$$R(k,-1)=0 \ ; \ R(k,0)=Y(k,0) \ ; \qquad (2.7)$$

$$R(k,m)=R(k+1,m-1)+[R(k+1,m-1)-R(k,m-1)]/\{(h_k/h_{k+m})^2[1-$$
$$(R(k+1,m-1)-R(k,m-1))/(R(k+1,m-1)-R(k+1,m-2))]-1\} \qquad (2.8)$$

for $m\geq1$. This scheme is illustrated (for $K=5$) in Figure 2. The process is stopped ([10]) when

$$\text{abs}(R(k-m,m)-R(k-m+1,m))<\text{tol} \qquad (2.9)$$

for some $K$ and $m$. If (2.9) holds then the result $R(k-m+1,m)$ could be accepted.

**2.2.** Basic pre-processing methods. In the case of non-stiff equations ([10], [13]), for r=1 the basic method to compute the $Y(k,0)$-values is the Euler method

$$y_{i+1}=y_i+h_kf(t_i,y_i), t_{i+1}=t_i+h_k, \ i=0,...,n_k-1, \qquad (2.10)$$
$$Y(k,0)=y_{nk}.$$

For $r=2$, a numerical integration formula for which (2.3) holds is the second order Gragg's method described by

$$z(t_0,h_k)=y_0, \qquad (2.11a)$$

$$z(t_1,h_k)=y_0+h_kf(t_0,y_0);$$

$$t_i=t_0+ih_k, \quad z(t_{i+1},h_k)=z(t_{i-1},h_k)+2h_kf(t_i,z(t_i,h_k)); \qquad (2.11b)$$

$$i=1,\dots,n_{k-1}:$$

$$Y(k,0)=[z(t_nk-1,h_k)+z(t_nk,h_k)+h_kf(t_nk,z(t_nk,h_k))]/2$$

Extrapolation of implicit methods can be used for stiff equations. Some symmetric and non-symmetric methods which are of interest are described in [12]. In this case, for each $Y(k,0)$-value, we must solve $n_k$ nonlinear equations, and consequently the computing time cannot be predicted. However, a lower bound to this time could be given by the time needed by Gragg's method. In what follows this lower bound will be used instead of the exact time.

**3. Parallel pre-processing algorithms.** This section deals with the parallel computation of $Y(k,0)$, $k=0,\dots,K$. We consider the case of explicit methods. The entire computational diagraph of the polynomial extrapolation is illustrated for $K=5$ in Figure 3. The computation required by $Y(k,0)$ is represented by the sequence of vertices denoted by seq($k$). Each vertex in this sequence represents just one step of the Euler method, while an arc $(v,v')$ denotes the sending of the $y$-value from $v$ to $v'$. The first vertex in each sequence requires $Y_0$ as the starting value.

For each fixed $k$, $Y(k,0)$ is obtained by performing $n_k$ steps with the step size $h_k$. It is reasonable to suppose that each step requires an amount of time which depends on the applied explicit method and the complexity in evaluating $f(t,y)$, and does not depend on the step size $h_k$.

Let $t(0)$ be the time to perform the operation $0\epsilon(+,*,/)$ and define the time unit $(TU)$ as $t(/)$. Let us denote by $c$ the time to execute just one step of the Euler's method (2.10). The dominant part of $c$ is represented by the time to compute $f$. In the case of formula (2.10), $Y(k,0)$ requires $cn_k$ $TUs$.

If Gragg's method is used then $Y(k,0)$ is computed in $c+(n_k-1)[c+t(+)]+(c+t(+)+t(*))=n_k+1$ $TUs$, because $t(+)<<c$. Therefore, the use of the Gragg's method for computing the initial $Y$-values, can be studied by considering $n_k=n_k+1$. In the case of $n_k=ak+b$ this can be simply done by taking $b:=b+1$. Some problems arise for $n_k=2^k$ and $n_{k+1}=n_k+n_{k-1}$. In each case, $n_k$ exponentially increases and $n_k$ does not differ significantly from $n_k$. So, we can reduce the discussion to Euler's method applied to the above step number sequences. Also, we can suppose that $c=1$.

We make use of the idealized model of parallel computation known as the PRAM ([11]). We shall suppose that $S$ processors, $P_s$, $s=1,\dots,S$, are available. We consider the parallel execution of the sequences seq$(k)$, $k=0,\dots,K$, where $K+1>S$. The processors execute the same program implementing Euler's method for the same function and initial condition, but for different step sizes and number of steps. We remark that the use of the PRAM model is motivated by the necessity to consider functions of arbitrary complexity and not by the intrinsec structure of the numerical method computing the $Y(k,0)$-values. We note that if $f$ is a polynomial both in $t$ and $y$, or is given by a small size arithmetic expression including elementary functions, then a soft/hard-systolic implementation is also possible ([2-6], [14-15], [17-18]).

**3.1.** Case $n_k=ak+b$. Let $Q$ and $Q_0$ be two positive integers so that $K+1=(2S)Q+Q_0$, with $Q_0<2S$.

The $K+1$ sequences are organized in $Q$ bands $B(q), q=1,...,Q$, of 2S succesive sequences and the band $B(Q+1)$ of $Q_0$ sequences. The bands $B(q)$, $q=1,...,Q+1$, are processed in a serial fashion, while the sequences of each band are processed in parallel, as it is illustrated in Figure 4 for $S=4$. The $q$-th band is formed by seq($(q-1)2S+j-1$), $j=1,...,2S$, $q=1,...,Q$.

The processors $P_1,...,P_S$ begin the execution of $B(q)$ at the same time, say ts($q$). $P_r$ executes seq($(q-1)2S+r-1$) and then seq($(q-1)2S+2S-r$), $r=1,...,S$, and this computation takes $Tp(q)=t((q-1)2s+r-1)+t((q-1)2s+2s-r)=a[4(q-1)s+2s-1]+2b$ $TUs$.

This time does not depend on $r$, thus the processors terminate the processing of $B(q)$ at the same time, $ts(q)+Tp(q)$. A serial algorithm takes $Ts(q)=S*Tp(q)$, then the efficiency of processor utilization is $Ec(q)=Ts(q)/(STp(q))=1$ i.e. the strategy to process each band is optional.

The execution of $B(q+1)$ starts at $ts(q+1)=ts(q)+Tp(q)$ and is done in the same manner. If $Q_0=0$, then the parallel processing time for $Q$ bands is

$$Tp=ts(1)+Tp(1)+...+Tp(Q)=Q[2b+a(2S-1)+2aS(Q-1)], \qquad (3.1)$$

for $ts(1)=0$. The total time required by a serial algorithm is

$$Ts=Ts(1)+\ldots+Ts(Q)+S[Tp(1)+\ldots+Tp(Q)]=S*Tp, \qquad (3.2)$$

therefore the global efficiency $Ec=1$.

Now, let us suppose that $Q\neq 0$ and $B(Q+1)$ is processed like the previous bands.

First, if $1\leq Q_0\leq S$ we use only $Q_0$ processors to process $\mathrm{seq}(2QS-1+j)$, $j=1,\ldots,Q_0$. Therefore

$$Tp'(Q+1)=t(2QS-1+Q_0)=t(K)=aK+b \text{ TUs},$$

while

$$Ts'(Q+1)=t(2QS)+t(2QS+1)+\ldots+t(2QS+Q_0-1)=$$
$$=[a(2QS-1)+b]Q_0+aQ_0(Q_0+1)/2.$$

Thus, we obtain that the effectiveness of processor utilization for the last band is

$$Ec'(Q+1)=1-a(Q_0-1)/[2(aK+b)].$$

On the other hand,

$$Tp'=Tp+Tp'(Q+1),$$
$$Ts'=Ts+Ts'(Q+1),$$

and from (3.1) and (3.2) we obtain the effectiveness of processor utilization for the entire process as

$$Ec'=Ts'/(STp')=$$
$$=1-[aQ_0^2/2-Q_0(aK+b+a/2)+S(aK+b)]/S(Tp+aK+b)]<1,$$

and $Ec'$ has the greatest value for $Q_0=S$ when

$$Ec'=1-a(S-1)/[2(Tp+aK+b)].$$

Second, if $S+1\leq Q_0\leq 2S-1$, we have that $S$ processor execute the sequences $\mathrm{seq}(2QS-1+j)$, $j=1,\ldots,Q_0$. Let us take $Q_\delta\epsilon$ $(1,\ldots,S-1)$, so that $Q_0=S+Q_\delta$. For the sake of regularity we use the same strategy for peocessing this last band. Therefore, $P_j$ performs only

seq$(2SQ-1+j)$, $j=1,2,\dots,S-Q_\delta$ while $P_h$ executes seq$(2QS-1+h)$ and then seq $(2QS+2S-h)$, $h=S-Q_\delta+1,\dots,S$. Consequently, the parallel processing time for the last band is

$$Tp''(Q+1)=t(2QS-1+S)+t(2QS+S)=a(4QS+2S-1)+2b \ TUs,$$

while the corresponding sequential time is

$$Ts''(Q+1)=Ts'(Q+1)$$

and we obtain that

$$Ec''(Q+1)=(Q_0/S)[a(K+1-Q_0-1)+b+a(Q_0+1)/2]/[a(2(K+1-Q_0)+2S-1)+2b].$$

In this case, the total parallel processing time is

$$Tp''=Tp+Tp''(Q+1),$$

and the corresponding sequential time is

$$Ts''=Ts+Ts''(Q+1),$$

and from (3.1) and (3.2), the obtained effectiveness of processor utilization for the entire process is

$$Ec''=Ts''/(S*Tp'')=1-\{a[-2S^2(1+2Q)+S(1+2QQ_0)+Q_0(Q_0-1)/2]+b(-2S+Q_0)\}/$$
$$[S[Tp+a(4QS+2S-1)+2b]\}.$$

As a conclusion of the above analysis we can state

THEOREM 1. *Under the above assumptions, if $n_k=ak+b$, $k=0,\dots K$ and $S$ processors are used to compute $Y(k,0)$, $k=0,\dots,K$, then the following assertions are true:*

(i) *if $K+1=2SQ$ then the algorithm is optional with respect to processor utilization and the parallel processing time is*

$$Tp=Q[2+a(2S-1)+2aS(Q-1)];$$

(ii) *if $Q_0=K+1-2SQ$ and $1\le Q_0\le S$ then the parallel processing time is*

$$Tp'=Tp+aK+b$$

*and the effectiveness of processor utilization is*

$Ec'=1-[aQ_0^2/2-Q_0(aK+b+a/2)+S(aK+b)]/[S(Tp+aK+b)]$

$\leq 1-a(S-1)/[2(Tp+aK=b)]$,

*while the equality holds for* $Q_0=S$;

(iii) *if* $Q_0=K+1-2SQ$ *and* $S+1\leq Q_0\leq 2S-1$ *then the parallel processing time is*

$$Tp''=Tp+a(4QS+2S-1)+2b,$$

*and the effectiveness of processor utilization is*

$Ec''=1-\{a[-2S^2(1+2Q)+S(1+2QQ_0)+Q_0(Q_0-1)/2]+b(-2S+Q_0))/$

$\{S[Tp+a(4QS+2S-1)+2b]\}$.

**The above discussion could be extended in the following way. Let $m$ be a positive integer, $1<m<K+1$, with $K+1=m*N$, and define $C_p=\{seq(p+jm)/j=0,...,N-1\}$, $p=0,...,m-1$. Also, consider the positive integers $a_p$ and $b_p$, $p=0,...,m-1$. A mixed pre-processing strategy is to compute $Y(k,0)$ in accordance to the step number sequence $a_p*k+b_p$, as it is required in the case of stiff equations ([12]). A desirable property of the parallel processing scheme is that whenever $k<k'$, the $Y(k,0)$-value is obtained before $Y(k'.0)$, because in this case the stopping condition (2.6) can** be efficiently used to save time and hardware. On the other hand, **the precedence constraints appearing in the extrapolation stage** do not require such a property (see Figures 1-2). It **results that we could relaxe the above condition by requiring that it holds** only for a given number of succesive $Y(k,0)$-values. So, an **acceptable compromise** is to **assign** seq($p+jm$), $p=0,...,m-1$, to **processor** $p_{j+1}$, $j=0.,,,.S-1$, $S>N$, **and to continue with** $P_1$,

$P_2, \ldots$ for $j=S, S+1, \ldots$ and so on. This solution is compatible with the necessity to adopt a local synchronization technique ([19], [16]) for interfacing the set of $S$ processors and the systolic arrays in [8].

**3.2.** Case $n_k=2^k$. If $S$ processors are available to compute seq($k$), $k=0,\ldots,K$, the obtained efficiency is

$$Ec(S)=N(K)/(S*Tp(S)),$$

where $N(k)=n_0+\ldots+n_k$ and $Tp(S)$ is the corresponding parallel processing time. Since $N(k)=2n_k-1$ and

$$Tp(S)\geq\max\{n_k/k=0,1,\ldots,K\}=n_K,$$

we obtain

$$Ec(S)=(2n_K-1)/(Sn_K)\leq(2n_K-1)/(2n_K).$$

This upper bound does not depend on $S$ and because

$$Ec(2)=(2n_K-1)/(2n_K)$$

we conclude that it is fruitless (from the efficiency point of view) to use more than two processors to execute seq($k$), $k=0,1,\ldots,K$. If the processors are $P_1$ and $P_2$, then $P_1$ executes seq($k$),$k=0,\ldots,K-1$ in $N(K-1)$ *TUs* and $P_2$ performs seq($K$) in $n_K=N(K-1)+1$ *TUs* as it is illustrated in Figure 5. If we need to extend the activity of $P_i$, $i=1,2$, to seq($k$), $k=K+1,\ldots,K+R+1$, and $P_1$ executes seq($k$), $k=K+1,\ldots,K+R$, while $P_2$ acts on seq($K+R+1$), then the obtained efficiency is

$$Ec(2)=n_{K+1}N(R)/(2n_{K+R+1})=1-1/2^{R+1}.$$

Now, $P_1$ works in $n_{K+1}N(R-1)=2^{K+R+1}-2^{K+1}$ *TUs*, while $P_2$ needs $n_{K+R+1}$ *TUs*.

These results suggest the following strategy. Let us suppose that $S=2S'$ and the sequence seq$(k)$, $k=0,\dots,K$ are divided into $M$ bands $B(m)$, $m=1,\dots,M$ of equal size, and band $B(m)$ is divided into $S'$ subbands $SB(m,s)$, $s=1,\dots,S$ of equal size $R$, i.e. $K+1=MS'R$. Therefore B(m) contains seq$((m-1)RS'+j-1)$, $j=1,\dots,RS'$, and $SB(m,s)$ consists of seq$((m-1)RS'+(s-1)R+r-1)$, $r=1,\dots,R$, $s=1,\dots,S'$, $m=1,\dots,M$. The processing begins with $B(1)$ so that the $S'$ pairs of processors start at the same time. The $s$-th pair of processors, say $(P(s,1),P(s,2))$, acts on $SB(1,s)$, $s=1,\dots,S'$ so that $P(s,1)$ executes the first $R-1$ sequences and $P(s,2)$ the $R$-th sequence. As soon as $P(s,j)$ terminates its work for $SB(m,s)$ it passes to the corresponding sequence(s) of $SB(m+1,s)$, and so on. This strategy is illustrated in Figure 6. It results that $P(s,1)$ executes seq$((m-1)RS'+(s-1)R+r-1)$, $r=1,\dots,R-1$, while $P(s,2)$ acts on seq$((m-1)RS'+sR-1)$, $m=1,\dots,M$, $s=1,\dots,S'$. Let $t_0$ be the time when $P(s,j)$, $s=1,\dots,S'$, $j=1,2$, start the activity for $B(1)$. Also, let us denote by $tf(k)$ $(tin(k))$ the time when the execution of $seq(k)$ is terminated (initiated). Clearly, for each $s\in\{1,\dots,S''\}$, from the activity of $P(s,2)$, we have that

$$tf(sR-1)=t_0+2^{sR-1},$$

$$tf((h-1)RS'+sR-1)=tf((h-2)RS'+sR-1)+2^{(m-1)RS'+sR-1}, \quad h=2,\dots,m,$$

and consequently, we obtain

$$tf((m-1)RS'+sR-1)=t_0+2^{sR-1}(2^{mRS'}-1)/(2^{RS'}-1), \quad m=1,\dots,M.$$

On the other hand,

$$tin((m-1)RS'+sR-1)=tf((m-1)RS'+sR-1)-2^{(m-1)RS'+sR-1}=$$
$$tf((m-2)RS'+sR-1), \quad m=1,\dots,M.$$

Now, let us analyse the activity of $P(s,1)$, $s\epsilon\{1,\dots,S'\}$. This processor executes in the order $((\text{seq}((m-1)RS'+(s-1)R+r-1), r=1,\dots,R-1), \quad m=1,\dots,M)$. Let $TB(s,m)$ be the time in which $P(s,1)$ executes the sequences belonging to $B(m)$, $m=1,\dots,M$. Clearly,

$$TB(s,m) = \sum_{r=1}^{R-1} 2^{(m-1)RS'+(s-1)R+r-1} =$$
$$= (2^{R-1}-1)\ 2^{(m-1)RS'+(s-1)R}$$

Also, let $TF(s,m)$ be the time when the last sequence of $B(m)$ associated to $P(s,1)$ is terminated. Clearly,

$$TF(s,1)=t_0+TB(s,1),$$
$$TF(s,h)+TF(s,h-1)+TB(s,h),\ h=1,\dots;m.$$

Thus,

$$TF(s,m) = t_0 + \sum_{h=1}^{m} TB(s,h)$$

$$t_0+2^{(s-1)}\ (2^{R-1}-1)(2^{mRS'}-1)/(2^{RS'}-1).$$

If $TS(s,m)$ denotes the time when $P(s,1)$ begins the execution of its first sequence from $B(m)$, then $TS(s,m)=TF(s,m)-TB(s,m)$.

For a better analysis, we compute $tf((m-1)RS'+(s-1)R+r-1)$, $r=1,\dots,R-1$, $m=1,\dots,M$. Clearly, we have while

$$tin((m-1)RS'+(s-1)R)=TS(s,m)$$

$$tf((m-1)RS'+(s-1)R+r-1)=TS(s,m)+\sum_{h=1}^{r} 2^{(m-1)RS'+(s-1)R+h-1} \tag{3.3}$$

$$=TS(s,m)+(2^r-1)2^{(m-1)RS'+(s-1)R},$$

and

$tin((m-1)RS'+(s-1)R+r-1)=tf((m-1)RS'+(s-1)R+r-2)$, $r=2,\ldots,R-1$.

Now, let us compute the effectiveness of processor utilization, $Ec(S',R)$.

The time $Ts$ required by a sequential algorithm is

$$Ts=n_0+\ldots+n_K=2^{K+1}.$$

The number $Np=2S'$ of processors leads to the parallel processing time $Tp$ whose value is obtained by analysing the activity of (i) $p(s,1)$, $s=1,\ldots,S'$ and (ii) $P(s,2)$, $s=1,\ldots,S'$.

(i) From (3.3) we obtain the equivalent form of the final time $tf((m-1)RS'+(s-1)R+r-1)=t_0+(2^{R-1}-1)(2^{mRS'}-1)2^{(s-1)R}/(2^{RS'}-1)-$

$$(2^{R-1}-1)2^{(m-1)RS'+(s-1)R}+$$
$$(2^r-1)2^{(m-1)RS'+(s-1)R},$$
$$r=1,\ldots,R-1,\ m=1,\ldots,M.$$

The last task executed by $P(s,1)$ is obtained for $m=M$ and $r=R-1$ and has the ending time $tf_{s,1}$ given by

$$tf_{s,1}=tf((M-1)RS'+(s-1)R+R-2)=$$
$$t_0+(2^{R-1}-1)(2^{K+1}-1)2^{(s-1)R}/(2^{RS'}-1),$$

because $K+1=MRS'$.

(ii) The last task executed by $P(s,2)$ has an end time $tf_{s,2}$ obtained from $tf((m-1)RS'+sR-1)$ by taking $m=M$, i.e. $tf_{s,2}=t_0+(2^{K+1}-1)2^{sR-1}/(2^{RS'}-1)$.

But

$Tp(S',R)=\max\{tf_{s,1},tf_{s,2}/s=1,\ldots,S'\}-t_0=\max\{tf1,tf2\}$, where

$$tf1=\max\{tf_{s,1}/s=1,\dots,S'\}-t_0=tf_{S',1}-t_0=$$
$$(2^{R-1}-1)(2^{K+1}-1)2^{(S'-1)R}/(2^{RS'}-1),$$

while

$$tf2=\max\{tf_{s,2}/s=1,\dots,S'\}-t_0=tf_{S',2}-t_0=$$
$$(2^{K+1}-1)2^{RS'-1}/(2^{RS'}-1),$$

and finally, we obtain

$$Tp(S',R)=(2^{K+1}-1)2^{RS'-1}/(2^{RS'}-1).$$

Therefore,

$$Ec(S',R)=Ts/(NpTp(S',R))=(1-1/2^{RS'})/S'. \qquad (3.4)$$

As it was expected, by taking $S'=1$ and $R=K+1$ in (3.4), it results

$$Ec(1,K+1)=1-1/2^{K+1}$$

and

$$Ec(1,K+1)\geq EC(S',R)S'.$$

Also, for $M=1$ it holds $Ec(1,K+1)=Ec(S',R)S'$.

On the other hand, $Tp(1,K+1)=2^K$ and

$$Tp(S',R)/Tp(1,K+1)=(1-1/2^{K+1})/(1-1/2^{(K+1)/M}). \qquad (3.5)$$

By taking $M=1$ in (3.5), $Tp(S',R)=Tp(1,K+1)$ results. If $M>1$ consider $f(x)=(1-x^M)/(1-x)$, with $0<x<1/2$. A simple calculation shows that $1<f(x)<2-1/2^{M-1}$ and therefore we obtain that

$$Tp(1,K+1)<Tp(S',R)<Tp(1,K+1)(2-1/2^{M-1}).$$

In fact, we have proved the following

THEOREM 2. *Under the above assumptions if $n_k=2^k$, $k=0,\dots,K$, and $S$ processors are used to compute $Y(k,0)$, $k=0,\dots,K$, then the following assertion are true:*

(i) *if $Ec(S)$ is the effectiveness of processor utilization* *for an arbitrary parallel algorithm then* $Ec(S) \le Ec(2)$, *for* $S \ge 2$;

(ii) *if* $S=2S'$ *and* $K+1=MRS'$ *and the above strategy is used then*

(ii.i) *the parallel processing time is*

$$Tp(S',R)=(2^{K+1}-1)2^{RS'-1}/(2^{RS'}-1);$$

(ii.ii) *the effectiveness of processor utilization is*

$$Ec(S',R)=(1-1/2^{RS'})/S';$$

(ii.iii) *if* $M=1$ *then*

$$Tp(1,K+1)=Tp(S',R)$$

*and*

$$Ec(1,K+1)=Ec(S',R)S';$$

(ii.iv) *if* $M>1$ *then*

$$Tp(1,K+1)<Tp(S',R)<Tp(1,K+1)(2-1/2^{M-1})$$

*and*

$$Ec(1,K+1)>Ec(S',R)S'.$$

**3.3. Case $n_{k+1}=n_k+n_{k-1}$. Let us consider the case when $(n_k)$ is a Fibonacci sequence, for some prescribed $n_0$ and $n_1$. The reason to consider this sequence is that it is possible to determine $a$, $b$, $n_0$ and $n_1$ such that $ak+b \le n_k \le 2^{**}k$ holds for $k \ge k_0$, where $k_0$ is a prescribed rank. We suppose again that $S=2S'$. We split the set of $K+1$ sequences into $S'$ bands $B(s)$, $s=1,\dots,S'$. The band $B(s)$ splits into the subbands $SB(s,r)$, $r=1,\dots,R$, where $SB(s,r)$ consists of three succesive sequences, i.e. $K+1=3RS'$ and $SB(s,r)=\{seq((s-1)3R+3(r-1)+h)/h=0,1,2\}$ as it is shown in Figure 7(a). The pair of processors $(P(s,1),P(s,2))$ acts on $B(s)$, $s=1,\dots,S'$ and all pairs begin the activity at the same time $t_0$**

(Figure 7(b)). The processor $P(s,1)$ executes the sequences $((seq((s-1)3R+3(r-1)), seq((s-1)3R+3(r-1)+1), r=1,...,R)$ in this order, while $P(s,2)$ executes $(seq((s-1)3R+3(r-1)+2), r=1,...,R)$ in this order. Therefore $P(s,1)$ needs $t(s,1)$ time, where

$$t(s,1) = \sum_{r=1}^{R} (n_{(s-1)3R+3(r-1)} + n_{(s-1)3R+3(r-1)+1}),$$

while the time required by $P(s,2)$ is

$$t(s,2) = \sum_{r=1}^{R} n_{(s-1)3R+3(r-1)+2}, \tag{3.6}$$

and because $(n_k)$ is a Fibonacci sequence it results that $t(s,1)=t(s,2)$. Therefore, the time to execute $B(s)$ is $t(s,2)$ and the parallel processing time is

$$Tp=\max\{t(s,2)/s=1,...,S'\}=t(S',2)=$$

$$\sum_{r=1}^{R} n_{(s'-1)3R+3r-1}TUs.$$

If $q_1$ and $q_2$ are the roots of the equation $x^2-x-1=0$, then

$$n_k=c_1q_1{}^k+c_2q_2{}^k, \tag{3.7}$$

where $c_j$, $j=1,2$, are determined by $n_0$ and $n_1$. Using this form of $n_k$ and taking into account that $q_j{}^3-1=2q_j$, $j=1,2$, from (3.6) we obtain

$$t(s,2) = \sum_{i=1}^{2} c_i \sum_{r=1}^{R} q_i^{(s-1)3R+3(r-1)+2} = \\ 0.5 \sum_{i=1}^{2} c_i q_i^{(s-1)3R+1} (q_i^{3R}-1). \tag{3.8}$$

Therefore, it is obtained that

$$Tp=0.5\sum_{i=1}^{2} c_i q_i^{(S'-1)3R+1}(q_i^{3R}-1)\,.$$

On the other hand, the time required by a sequential algorithm is

$$Ts = 2\sum_{s=1}^{S'} t(s,2)\,,$$

and from (3.8) we produce

$$Ts = \sum_{i=1}^{2} c_i q_i (q_i^{K+1}-1)\,,$$

while from (3.7) and the fact that $K+1=3RS'$, we obtain

$$Tp=0.5(n_{K+2}-n_{K+2-3R})$$

and

$$Ts=n_{K+2}-n_1\,.$$

Thus, the effectiveness of processor utilization is

$$Ec=Ts/(S*Tp)=3R(n_{K+2}-n_1)/[(K+1)(n_{K+2}-n_{K+2-3R})]\,.$$

Let us remark that $S'=1$ yields $k+1=3R$ and $n_{K+2-3R}=n_1$, consequently $Ec=1$.

As a consequence of the above analysis we can state

THEOREM 3. *Under the above assuptions, if $\{n_k\}$ is a Fibonacci sequence, a number of $S=2S'$ processors are used to compute $Y(k,0)$, $k=0,\dots,K$, and the above parallel algorithm is used, then:*

(i) *the parallel processing time is*

$$Tp=0.5(n_{K+2}-n_{K+2-3R})\ TUs;$$

(ii) *the effectiveness of processor utilization is*

$$Ec=3R(n_{K+2}-n_1)/[(K+1)(n_{K+2}-n_{K+2-3R})];$$

(iii) *if* $S=2$ *then the parallel algorithm is optimal with respect to the processor utilization.*

Now, we are able to give a better motivation to our work. A serial implementation of the extrapolation stage requires $O(K^2)$ amount of time. The pre-processing stage needs $O(cK^2)$ time for $n_k=ak+b$, and $O(cq^K)$ time, where $q=2$ for $n_k=2^k$ and $q=\max(q_1,q_2)$ for Fibonacci sequence, where $c$ includes the time complexity in the evaluation of $f$. The comparison still indicates that a parallel approach of the pre-processing stage is well suited. A similar situation appears in the case of Romberg's extrapolation method for numerical integration ([1], [7]).

**4. Final remarks.** There are two contradictory aspects in any attempt to parallelize the extrapolation methods for solving initial value problema in ODE's.

The first aspect refers to the adaptive feature of the method which allows us to stop the first stage computation as soon as it is obtained the desired accuracy. A considerable amount of time can be saved in this way. Clearly, the best way to accomplish the adaptive task is to use a serial algorithm, which starts the computation of $Y(k,0)$ value only if $Y(h,0)$, $h=0,\ldots,k-1$, do not lead to a true value of the convergence test.

The second aspect concerns the fact that if we wish to obtain a short parallel computing time, then the parallel algorithm must anticipate the computation of some $Y(k,0)$-values before knowing that these values are necessary or not to obtain the desired tolerance. If they aren't then an useless computation was performed, and this comes as a price. In compensation, a certain gain of accuracy could be obtained if the extrapolation stage continues while it consumes the $Y(k,0)$ values whose computation was already started and does not need much time to be finished.

The proposed algorithms accomplished a serial processing of the bands, while the parallel processing addresses to the tasks within of each band. For this reason, they seem to be a good compromise between opposing restrictions.

## R E F E R E N C E S

1. Brudaru, O., *Systolic arrays for numerical integration with Romberg's formula*, Analele Ştiinţifice ale Universităţii "Al. I. Cuza" din Iaşi, Informatica, no. 35, 1989, pp. 367-374.
2. Corbaz, G., Duprat, J., Hochet, B., Muller, J.M.,*Implementation of VLSI polynomial evaluator for real-time applications*, RR 91-07, LIP, ENSL, France, 1991.
3. Darte, A., Risset, T., Robert, Y., *Synthetizing systolic arrays: some recent developments*, RR 91-09, LIP, ENSL, France, 1991.
4. Duprat, J., Muller, J.-M., *Evaluation of polynomials and elementary functions by integrated circuits*, RR 698-I, TIM3, IMAG, France, 1988.
5. Duprat, J., Muller, J.-M., *Hardwired polynomial evaluation*, Journal of Parallel and Distributed computing, no. 5, 1988, pp. 291-309.
6. Evans, D. J., Margaritis, K., Bekakos, M.P., *Systolic and holographic pyramidal soft-systolic designs for succesive matrix powers*, Parallel Computing, no. 9, 1989, pp. 373-387.
7. Evans, D.J., Megson, G.M., *Romberg integration on systolic arrays*, Parallel Computing, no. 3, 1986, pp. 289-304.
8. Evans, D.J., Megson, G.M., *Construction of Extrapolation Tables by Systolic Arrays for Solving Ordinary Differential Equations*, Parallel Computing, 410.1, 1987, pp. 33-48.
9. Fisher, A.L., Kung, H.T., *Synchronizing large VLSI processor arrays*, IEEE Transactions on Computers, vol. 34, no. 8, 1985, pp. 734-740.
10. Gear, W.G., *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Inc., Englewood Cliff, New Jersey, 1971.
11. Gibons, A., Rytter, W., *Efficient Parallel Algorithms*, Cambridge University Press, Cambridge, 1988.
12. Hairer, E., Wauner, G., *Solving Ordinary Differential Equations II-Stiff and Differential-Algebraic Problems*, Springer Verlag, Berlin, 1991.
13. Jain, M.K., *Numerical Solution of Differential Equation*, Wiley Eastern Limited, New Delhi, 1984.
14. Kung, H.T., *Let's Design Algorithms for VLSI Systems*, Technical Report CMU-CS-79-151, Carnegie-Mellon University, Computer Science Departament, Pittsburgh, January, 1979.
15. Kung, H.T., *Why Systolic Architectures?*, Computer, 15, 1982, pp. 37-46.

16. O'Leary, D.P., Stewart, G.W., *From Determinancy to Systolic Arrays,*IEEE Transactions on Computers, vol. C-36, no. 11, November, 1987.
17. Privat, G., Renaudin, M., *L'algorithme CORDIC dans les architectures specialisees de traitement numerique du signal,* Traitement du Signal, vol. 5, no. 6, 1988, pp. 421-434.
18. Rajopadhye, S.V., Fujimoto, R.M., *Synthesizing Systolic Arrays from Recurrence Equations,* Parallel Computing, no.14, 1990, pp. 163-189.

$$\begin{tikzcd}
y(t,h_0)=Y(0,0) \arrow[dr] & & & & & \\
& Y(0,1) \arrow[dr] & & & & \\
y(t,h_1)=Y(1,0) \arrow[ur] \arrow[dr] & & Y(0,2) \arrow[dr] & & & \\
& Y(1,1) \arrow[ur] \arrow[dr] & & Y(0,3) \arrow[dr] & & \\
y(t,h_2)=Y(2,0) \arrow[ur] \arrow[dr] & & Y(1,2) \arrow[ur] \arrow[dr] & & Y(0,4) \arrow[dr] & \\
& Y(2,1) \arrow[ur] \arrow[dr] & & Y(1,3) \arrow[ur] \arrow[dr] & & Y(0,5) \\
y(t,h_3)=Y(3,0) \arrow[ur] \arrow[dr] & & Y(2,2) \arrow[ur] \arrow[dr] & & Y(1,4) \arrow[ur] & \\
& Y(3,1) \arrow[ur] \arrow[dr] & & Y(2,3) \arrow[ur] & & \\
y(t,h_4)=Y(4,0) \arrow[ur] \arrow[dr] & & Y(3,2) \arrow[ur] & & & \\
& Y(4,1) \arrow[ur] & & & & \\
y(t,h_5)=Y(5,0) \arrow[ur] & & & & &
\end{tikzcd}$$

**Figure 1:** Polynomial extrapolation data dependencies.

$$\begin{tikzcd}
 & Y(0,0)=R(0,0) \arrow[dr] & & & & & \\
R(1,-1)=0 \arrow[rr] & & R(0,1) \arrow[dr] & & & & \\
 & Y(1,0)=R(1,0) \arrow[ur] \arrow[dr] \arrow[rr] & & R(0,2) \arrow[dr] & & & \\
R(2,-1)=0 \arrow[rr] & & R(1,1) \arrow[ur] \arrow[dr] \arrow[rr] & & R(0,3) \arrow[dr] & & \\
 & Y(2,0)=R(2,0) \arrow[ur] \arrow[dr] \arrow[rr] & & R(1,2) \arrow[ur] \arrow[dr] \arrow[rr] & & R(0,4) \arrow[dr] & \\
R(3,-1)=0 \arrow[rr] & & R(2,1) \arrow[ur] \arrow[dr] \arrow[rr] & & R(1,3) \arrow[ur] \arrow[dr] \arrow[rr] & & R(0,5) \\
 & Y(3,0)=R(3,0) \arrow[ur] \arrow[dr] \arrow[rr] & & R(2,2) \arrow[ur] \arrow[dr] \arrow[rr] & & R(1,4) \arrow[ur] & \\
R(4,-1)=0 \arrow[rr] & & R(3,1) \arrow[ur] \arrow[dr] \arrow[rr] & & R(2,3) \arrow[ur] & & \\
 & Y(4,0)=R(4,0) \arrow[ur] \arrow[dr] \arrow[rr] & & R(3,2) \arrow[ur] & & & \\
R(5,-1)=0 \arrow[rr] & & R(4,1) \arrow[ur] & & & & \\
 & Y(5,0)=R(5,0) \arrow[ur] & & & & &
\end{tikzcd}$$

**Figure 2:** Rational extrapolation data dependencies.

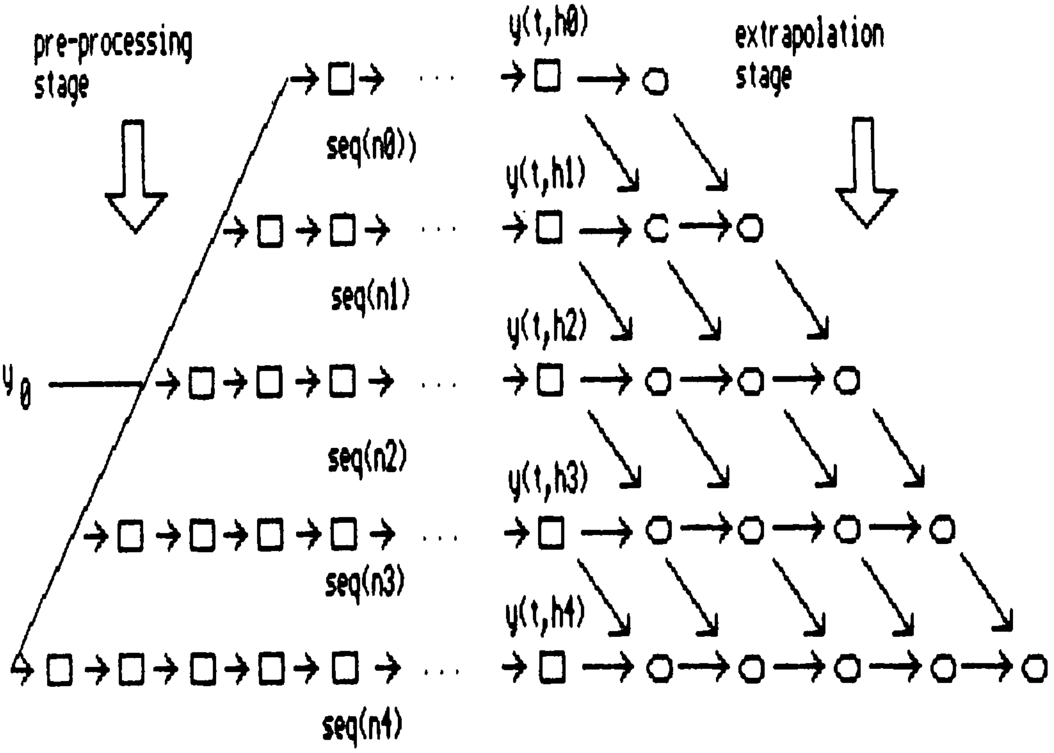


**Figure 3:** Polynomial extrapolation including the pre-processing stage (K=5).

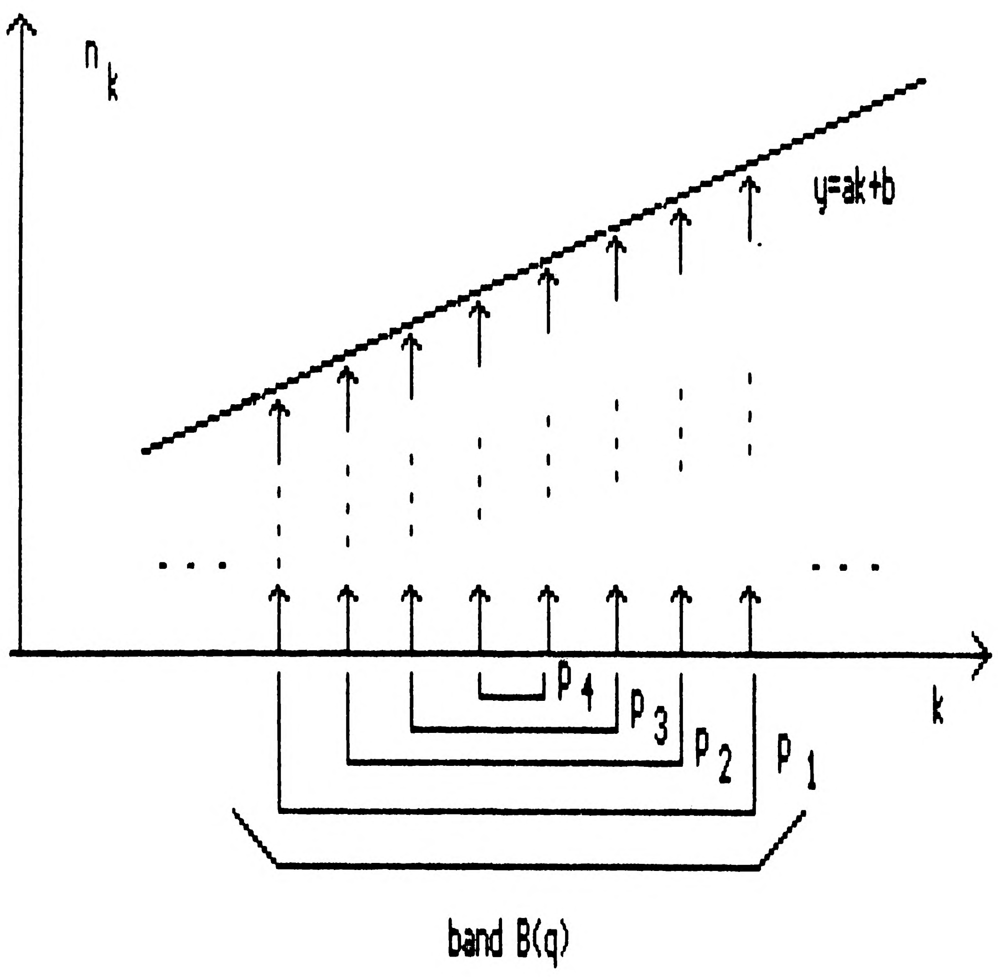


**Figure 4:** Parallel pre-processing for $n_k$=$ak+b$.

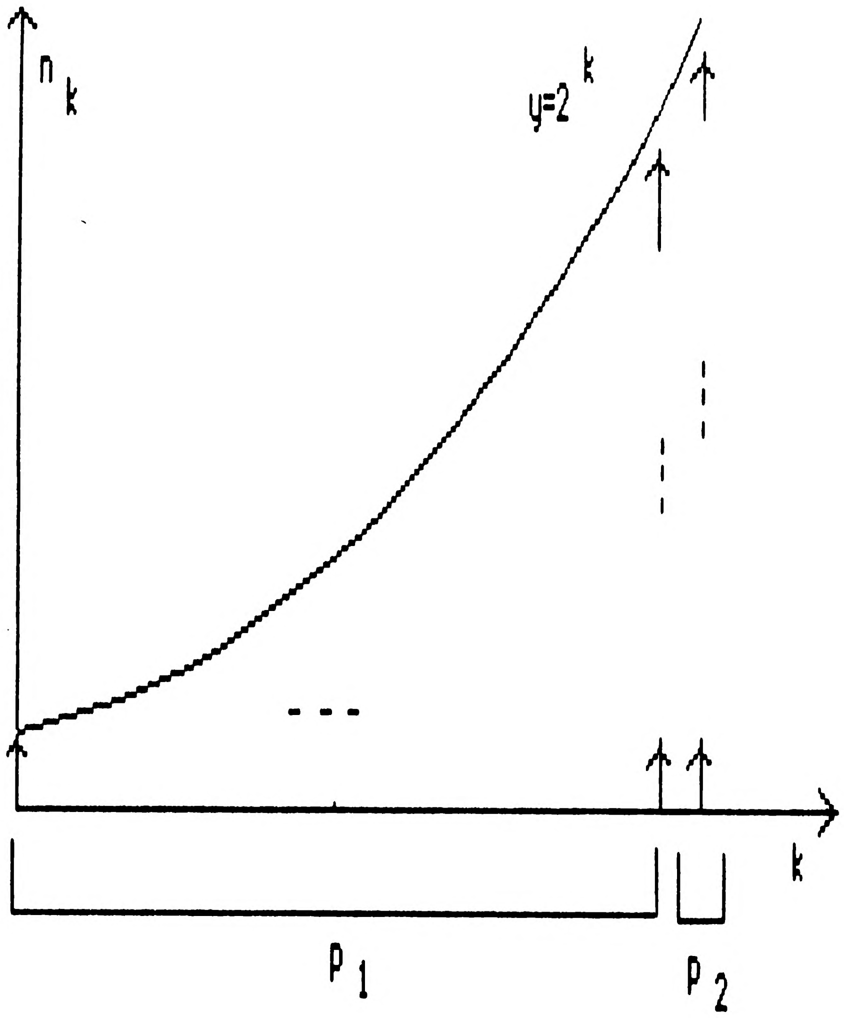


**Figure 5:** Two processors executing the pre-processing stage for $n_k$=2**$k$.

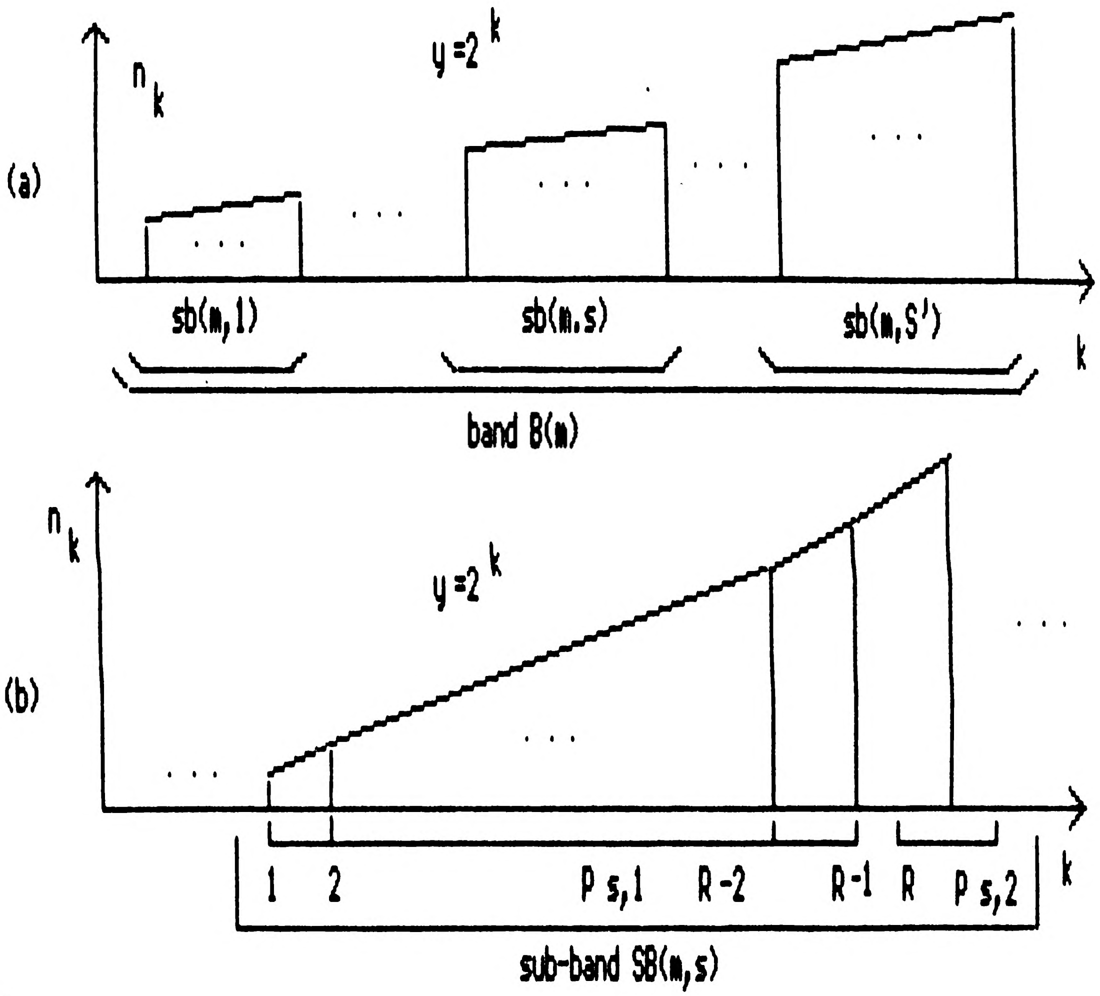
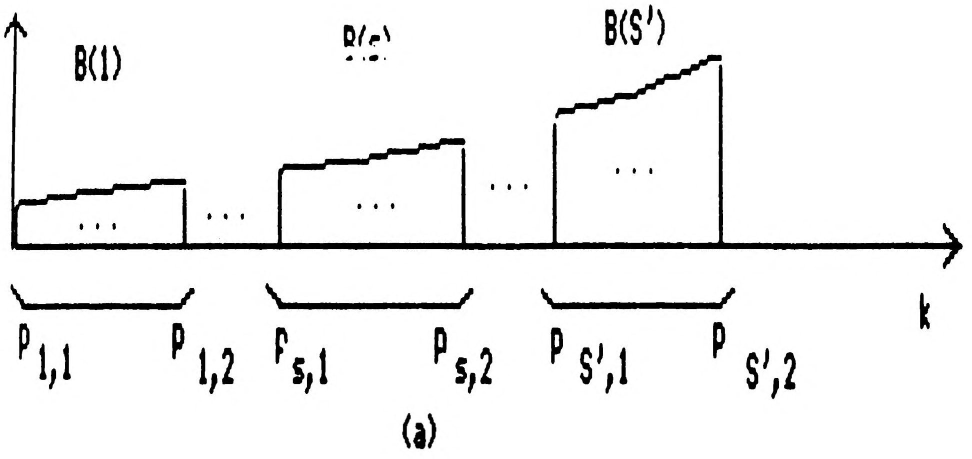


**Figure 6:** Parallel pre-processing for $n_k=2**k$:
(a) band structure of the sequence;
(b) the execution of $SB(m,s)$ by the pair $(P(s,1), P(s,2))$.



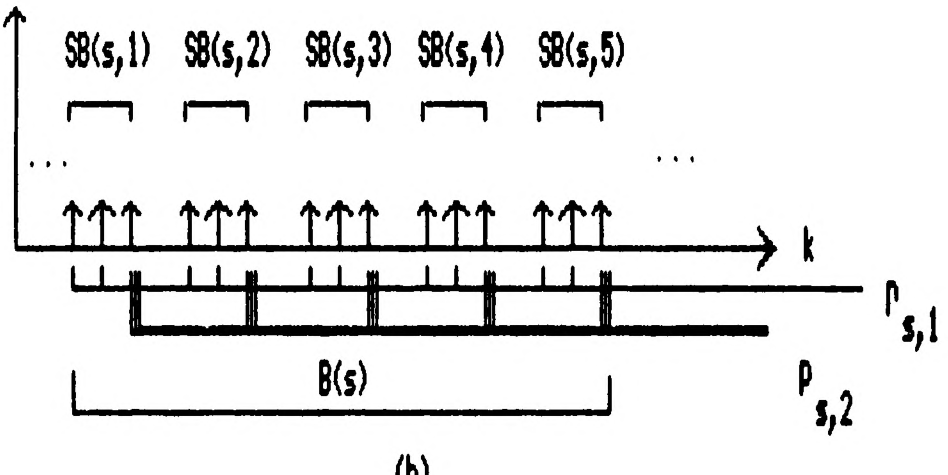


**Figure 7:** (a) The band structure of the $K$+1 sequences
(b) The sequences assigned to $P(s,1)$ and $P(s,2)$ ($R$=5) when $n_{k+1}=n_k+n_{k-1}$.