ANNALES
UNIVERSITATIS MARIAE CURIE – SKŁODOWSKA
LUBLIN – POLONIA

VOL. LVI, 9 SECTIO A 2002

PRZEMYSŁAW STPICZYŃSKI

Efficient data–parallel algorithms for computing trigonometric sums

Abstract. In this paper new parallel versions of Goertzel and Reinsch algorithms for calculating trigonometric sums are introduced. To achieve portability, the parallel algorithms have been implemented in High Performance Fortran and can be run on variety of parallel computers. The experimental results on a cluster of Pentium II computers with PVM3 and ADAPTOR compilation system are presented. Efficiency of the parallel Reinsch algorithm is about eighty percent.

1. Introduction. Let us consider the problem of computing sums

$$C(x)=\sum_{k=0}^{n} b_k\cos kx \quad\text{and}\quad S(x)=\sum_{k=1}^{n} b_k\sin kx\ . \tag{1}$$

which is very important for some numerical algorithms [9]. For example, in trigonometric interpolation the sums (1) are used for computing values of trigonometric polynomials

$$p(x)=\beta_0+\beta_1 e^{ix}+\ldots+\beta_{N-1}e^{(N-1)ix}.$$

1991 *Mathematics Subject Classification.* 65Y05, 42A05.

Key words and phrases. Trigonometric sums, parallel algorithms, linear recurrence systems, High Performance Fortran, PVM.

The solution of the problem (1) can be found using well known sequential Goertzel and Reinsch algorithms which reduce to the problem of solving linear recurrence systems. The Reinsch algorithm is more complicated but has better numerical properties [9]. In [12, 7] we presented parallel versions of Goertzel and Reinsch algorithms based on the recently developed parallel algorithms for solving linear recurrence systems [10, 11, 6, 5]. However, our experiments on a Sequent Symmetry parallel computer with shared memory showed that the algorithms had a rather poor efficiency. Thus, the purpose of this paper is to present improvements of the parallel algorithms. Studying the special structure of linear recurrence systems formed according to the Goertzel and Reinsch algorithms, we show how to develop new parallel algorithms with better potential parallelism. To achieve portability, we have implemented the algorithms in High Performance Fortran [3], so they can be efficiently run on a wide variety of parallel computers with shared or distributed memory. Our parallel programs have been tested on a cluster of Pentium II computers running under Linux operating system with PVM3 [4] and ADAPTOR compilation system [1, 2].

2. Parallel algorithms.

In this section we briefly describe parallel algorithm for solving linear recurrence systems with constant coefficients and parallel versions of Goertzel and Reinsch algorithms.

2.1. Linear recurrence systems.

Let us consider the following formula

$$x_k = \begin{cases} 0 & \text{for } k \le 0 \\ f_k + \sum_{j=1}^{m} a_j x_{k-j} & \text{for } 1 \le k \le n \end{cases} \tag{2}$$

called linear recurrence system with constant coefficients. It can be efficiently solved in parallel using a recently developed algorithm [10, 11, 6, 5]. The idea of the algorithm is to rewrite (2) as the block system of linear equations

$$\begin{pmatrix} L & & & \\ U & L & & \\ & \ddots & \ddots & \\ & & U & L \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_p \end{pmatrix} = \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_p \end{pmatrix}, \tag{3}$$

where for $q = n/p > m$ we have

$$L = \begin{pmatrix} 1 & & & & & \\ -a_1 & \ddots & & & & \\ \vdots & \ddots & \ddots & & & \\ -a_m & & \ddots & \ddots & & \\ & \ddots & & \ddots & \ddots & \\ & & -a_m & \cdots & -a_1 & 1 \end{pmatrix} \in \mathbb{R}^{q\times q}$$

$$U = \begin{pmatrix} & -a_m & \cdots & -a_1 \\ & & \ddots & \vdots \\ & & & -a_m \\ 0 & & & \end{pmatrix} \in \mathbb{R}^{q\times q}.$$

Then we get the following recurrence system

$$\text{(4)} \qquad \begin{cases} \mathbf{x}_1 = L^{-1}\mathbf{f}_1 \\ \mathbf{x}_j = L^{-1}\mathbf{f}_j - L^{-1}U\mathbf{x}_{j-1} \quad \text{for } j = 2, \dots, p. \end{cases}$$

To solve this system let us consider the structure of the matrix

$$\text{(5)} \qquad U = -\sum_{k=1}^{m}\sum_{l=k}^{m} a_{m+k-l}\mathbf{e}_k\mathbf{e}_{q-m+l}^T,$$

where $\mathbf{e}_k$ denotes k-th unit vector of $\mathbb{R}^q$. Thus the equation (4) reduces to the form

$$\text{(6)} \qquad \begin{cases} \mathbf{x}_1 = L^{-1}\mathbf{f}_1 \\ \mathbf{x}_j = L^{-1}\mathbf{f}_j + \displaystyle\sum_{k=1}^{m} \alpha_j^k \mathbf{y}_k \quad \text{for } j = 2, \dots, p, \end{cases}$$

where $L\mathbf{y}_k = \mathbf{e}_k$ and $\alpha_j^k = \sum_{l=k}^{m} a_{m+k-l}x_{(j-1)q-m+l}$. Note that to compute vectors $\mathbf{y}_k$ we need to find only the solution of the system $L\mathbf{y}_1 = \mathbf{e}_1$, namely $\mathbf{y}_1 = (1, y_2, \dots, y_q)^T$. Then we form vectors $\mathbf{y}_k$ as follows

$$\text{(7)} \qquad \mathbf{y}_k = (\underbrace{0, \dots, 0}_{k-1}, 1, y_2, \dots, y_{q-k+1})^T.$$

This yields that the number of subsystems we must solve in parallel does not depend on the order of the system.

2.2. Goertzel algorithm. First let us observe that we can restrict our attention to the case where $x \neq k\pi$. If $x = k\pi$ then $S(x) = 0$ for all x and $\cos kx = \pm 1$, thus $C(x)$ can be computed using a simple summation algorithm. In case of the Goertzel algorithm [9] for finding (1), we need to compute two last components (namely S_1 and S_2) of the following linear recurrence system with constant coefficients:

$$S_k = \begin{cases} 0 & \text{for } k = n+1, n+2 \\ b_k + 2S_{k+1}\cos x - S_{k+2} & \text{for } k = n, n-1, \dots, 1 \end{cases} \tag{8}$$

and then we compute

$$C(x) = b_0 + S_1 \cos x - S_2 \text{ and } S(x) = S_1 \sin x. \tag{9}$$

To apply the formula (2) and then the parallel algorithm based on (6), let us observe that (8) can be rewritten as (2) with $m = 2$, $a_1 = 2\cos x = c$, $a_2 = -1$ and right–hand side coefficients given by

$$f_k = b_{n-k+1} \quad \text{for } k = 1, \dots, n. \tag{10}$$

Thus, for $\mathbf{z}_j = L^{-1}\mathbf{f}_j$, the formula (2) will be of the form

$$\begin{cases} \mathbf{x}_1 = \mathbf{z}_1 \\ \mathbf{x}_j = \mathbf{z}_j + (-x_{(j-1)q-1} + c x_{(j-1)q})\mathbf{y}_1 - x_{(j-1)q}\mathbf{y}_2 \quad \text{for } j = 2, \dots, p. \end{cases} \tag{11}$$

When we set

$$M = \begin{pmatrix} -y_{q-1} & c y_{q-1} - y_{q-2} \\ -y_q & c y_q - y_{q-1} \end{pmatrix} \in \mathbb{R}^{2\times 2} \tag{12}$$

then from (11) we get the following recurrence formula for two last entries of each vector $\mathbf{x}_j$

$$\mathbf{x}_j^{''} = \mathbf{z}_j^{''} + M\mathbf{x}_{j-1}^{''} \text{ for } j = 2, \dots, p.$$

Finally, we get

$$\begin{pmatrix} S_2 \\ S_1 \end{pmatrix} = \mathbf{x}_p^{''} = \mathbf{z}_p^{''} + \sum_{j=1}^{p-1} M^{p-j}\mathbf{z}_j^{''}. \tag{13}$$

Now let us observe that in fact there is no need to solve the system $L\mathbf{y}_1 = \mathbf{e}_1$. Indeed, to form (13) we only need three last entries of $\mathbf{y}_1$.

Proposition 1. *Let* $\mathbf{y} = (1, y_2, \dots, y_q)^T$ *be the solution of the system* $L\mathbf{y} = \mathbf{e}_1$*. Then for* $x \neq k\pi$*,* $k \in \mathbb{Z}$*, we have*

$$y_r = \frac{\sin rx}{\sin x} \tag{14}$$

where $r = q-2,\ q-1,\ q$*.*

Proof. Clearly, we have

$$\sin qx = \sum_{k=1}^{q} c_k \sin kx$$

where $c_k = 1$ for $k = q$ and $c_k = 0$ for $k = 1, \dots, q-1$. From (8) we obtain that $\sin qx = S_1 \sin x$, where S_1 is the last component of the solution of the system

$$\begin{pmatrix} 1 & & & \\ -2\cos x & 1 & & \\ 1 & \ddots & \ddots & \\ & 1 & -2\cos x & 1 \end{pmatrix} \begin{pmatrix} S_q \\ S_{q-1} \\ \vdots \\ S_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

so $y_q = S_1$. This yields that $y_q = \sin(qx)/\sin x$. Analogously we prove (14) for $r = q-2,\ q-1$. $\square$

Using the above proposition we can formulate the parallel version of the Goertzel algorithm:

G1. Solve in parallel $L\mathbf{z}_j = \mathbf{f}_j$, $j = 1, \dots, p$

G2. Using (12) and (14) form the matrix M

G3. Using (13) compute S_1 and S_2

G4. Using (9) compute $C(x)$ and $S(x)$.

2.3. Parallel Reinsch algorithm.

To avoid the influence of rounding errors on the final computed solution when x is close to 0, in Reinsch algorithm [9] we set $S_{n+2} = D_{n+1} = 0$ and if $\cos x > 0$, then we solve

$$\begin{cases} S_{k+1} = D_{k+1} + S_{k+2} \\ \ \ D_k = b_k + \beta S_{k+1} + D_{k+1} \end{cases} \tag{15}$$

for $k = n, n-1, \dots, 0$, where $\beta = -4\sin^2\frac{x}{2}$. If $\cos x \leq 0$, then we solve

$$\begin{cases} S_{k+1} = D_{k+1} - S_{k+2} \\ \ \ D_k = b_k + \beta S_{k+1} - D_{k+1} \end{cases} \tag{16}$$

where $\beta = 4\cos^2\frac{x}{2}$. Finally, we compute

$$S(x) = S_1 \sin x \text{ and } C(x) = D_0 - \frac{\beta}{2} S_1. \tag{17}$$

Now let us observe that the solution of (15–16) is equivalent to the solution of the following system of linear equations

$$L\mathbf{x} = \mathbf{f}, \tag{18}$$

where $\mathbf{x}, \mathbf{f} \in \mathbb{R}^{2n}$,

$$x_k = \begin{cases} S_{n-\lfloor k/2 \rfloor} & \text{for } k = 1, 3, \ldots, 2n-1 \\ D_{n-k/2} & \text{for } k = 2, 4, \ldots, 2n \end{cases}, \tag{19}$$

$$f_k = \begin{cases} b_n & \text{for } k = 1 \\ b_{n-1} - \delta b_n & \text{for } k = 2 \\ 0 & \text{for } k = 3, 5, \ldots, 2n-1 \\ b_{n-k/2} & \text{for } k = 4, 6, \ldots, 2n \end{cases} \tag{20}$$

and

$$L = \begin{pmatrix} 1 &&&&&& \\ -\beta & 1 &&&&& \\ \delta & -1 & 1 &&&& \\ & \delta & -\beta & 1 &&& \\ && \delta & -1 & 1 && \\ &&& \ddots & \ddots & \ddots & \\ &&&& \delta & -\beta & 1 \end{pmatrix} \in \mathbb{R}^{2n \times 2n} \tag{21}$$

with

$$\delta = \begin{cases} -1 & \text{for } \cos x > 0 \\ 1 & \text{for } \cos x \le 0 \end{cases} \tag{22}$$

To find the solution of (18) in parallel, let us observe that the system is of the form (3). Thus, applying our parallel algorithm form solving linear recurrence systems, we have to solve the following systems of linear equations

$$L^{(q)}\mathbf{z}_j = \mathbf{f}_j, \ \ j = 1, \ldots, p,$$

where $L^{(q)} \in \mathbb{R}^{2q \times 2q}$ is a submatrix of L built from its first $2q$ rows and columns. Next we need to form the matrix

$$M = \begin{pmatrix} y_{q-1}^{(1)} & y_{q-1}^{(2)} \\ y_q^{(1)} & y_q^{(2)} \end{pmatrix} \in \mathbb{R}^{2 \times 2} \tag{23}$$

using two last entries of vectors $\mathbf{y}_1 = (y_1^{(1)}, \ldots, y_q^{(1)})^T$ and $\mathbf{y}_2 = (y_1^{(2)}, \ldots, y_q^{(2)})^T$ given by

$$L^{(q)}\mathbf{y}_1 = (\delta, 0, \ldots, 0)^T \tag{24}$$

$$L^{(q)}\mathbf{y}_2 = (-1, \delta, 0, \ldots, 0)^T \tag{25}$$

and then compute S_1 and D_0 applying

$$\begin{pmatrix} S_1 \\ D_0 \end{pmatrix} = \mathbf{x}_p'' = \mathbf{z}_p'' - \sum_{j=1}^{p-1} M^{p-j}\mathbf{z}_j''. \tag{26}$$

Analogously to the Goertzel algorithm we can state the following propositions.

Proposition 2. *For $x \neq k\pi$, $k \in \mathbb{Z}$, two last entries of $\mathbf{y}_1 = (y_1^{(1)}, \ldots, y_q^{(1)})^T$ given by (24) satisfy*

$$y_{q-1}^{(1)} = (\delta \sin qx + \sin(q-1)x) \,/\, \sin x \tag{27}$$

$$y_q^{(1)} = \delta \cos qx + \cos(q-1)x + \frac{\beta}{2} y_{q-1}^{(1)} \tag{28}$$

where δ is defined by (22) and $\beta = -4\sin^2\frac{x}{2}$ for $\cos x > 0$ and $\beta = 4\cos^2\frac{x}{2}$ for $\cos x \leq 0$.

Proof. Finding two last entries of the solution of (24) is equivalent to the Reinsch algorithm for computing sums (1). Due to (20), the coefficients b_k, $k = 1, \ldots, q$, are given by

$$b_k = \begin{cases} \delta & \text{for } k = q \\ 1 & \text{for } k = q-1 \\ 0 & \text{for } k = 1, \ldots, q-2. \end{cases}$$

Thus, using the definition of the Reinsch algorithm we get

$$S(x) = S_1 \sin x = \sin(q-1)x + \delta \sin qx.$$

Analogously

$$C(x) = D_0 - \frac{\beta}{2} = \cos(q-1)x + \delta \cos qx.$$

Assigning $y_{q-1}^{(1)} = S_1$ and $y_q^{(1)} = D_0$ we get (27) and (28). $\square$

Proposition 3. *For $x \neq k\pi$, $k \in \mathbb{Z}$, two last entries of $\mathbf{y}_2 = (y_1^{(2)}, \ldots, y_q^{(2)})^T$ given by (25) satisfy*

$$y_{q-1}^{(2)} = -\sin qx / \sin x \tag{29}$$

$$y_q^{(2)} = -\cos qx + \frac{\beta}{2} y_{q-1}^{(2)} \tag{30}$$

where $\beta = -4\sin^2\frac{x}{2}$ for $\cos x > 0$ and $\beta = 4\cos^2\frac{x}{2}$ for $\cos x \leq 0$.

Proof. As in the proof of the previous proposition, finding two last entries of the solution of (25) is equivalent to the Reinsch algorithm for computing sums (1), where coefficients b_k, $k = 1, \ldots, q$, are given by

$$b_k = \begin{cases} -1 & \text{for } k = q \\ 0 & \text{for } k = 1, \ldots, q-1, \end{cases}$$

so $S_1 = -\sin qx / \sin x$ and $D_0 = \frac{\beta}{2} - \cos qx$. $\square$

Using the above propositions we can conclude that the parallel version of the Reinsch algorithm consists of the following steps:

R1. Solve in parallel $L^{(q)}\mathbf{z}_j = \mathbf{f}_j$, $j = 1, \ldots, p$

R2. According to (23) form the matrix M

R3. Using (26) compute S_1 and D_0

R4. Using (17) compute $S(x)$ and $C(x)$.

3. Data parallel implementation and experiments.

The algorithms have been implemented in High Performance Fortran [3]. The model of data parallel programming with HPF is based on the following principles:

- a user must define data mapping onto a virtual array of processors;
- a user must specify data parallelism using array operations, `FORALL` statements or an `INDEPENDENT` directive which asserts to a compiler that statements in a loop can be executed independently.

Let us briefly describe specific HPF constructions used in our implementation.

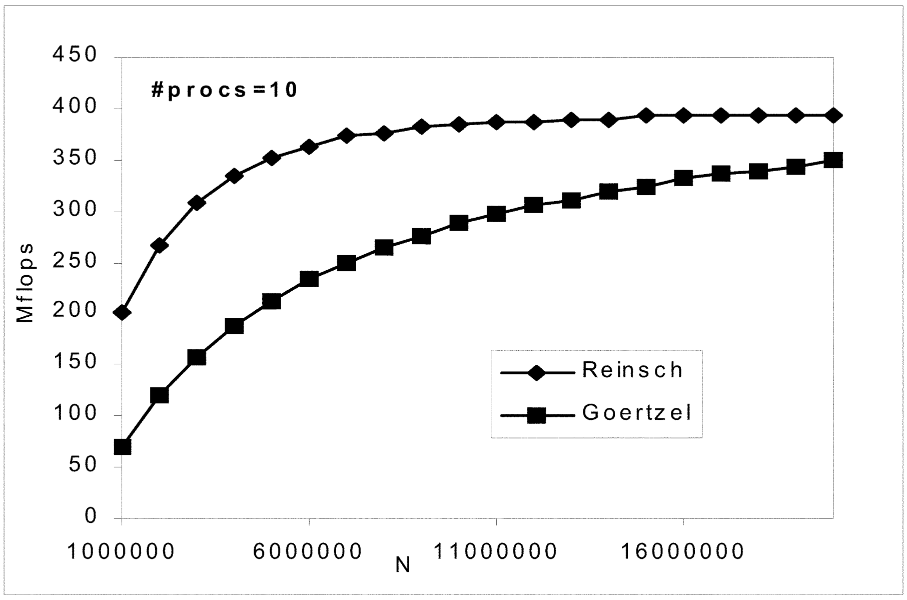

FIGURE 1: Performance for number of processors equal to 10

First of all we have to distribute coefficients f_k defined by (20) for Reinsch and (10) for Goertzel and then distribute them onto a virtual array of processors. Note, that for Reinsch we need to distribute only non–zero numbers, i.e. such that $k \neq 3, 5, \ldots, 2n-1$. For both of the algorithms, it can be done by the following HPF directives

```
      real*8, allocatable :: f(:,:), x(:,:)
!hpf$ processors P(number_of_processors())
!hpf$ distribute f(*,block) onto P
!hpf$ distribute x(*,block) onto P
!  .....
!  set p as the number of processors
!  set q as n/p
!  and then allocate arrays
      allocate (f(1:q,1:p),x(1:2,1:p))
```

The first step of the algorithms (i.e. solving in parallel p systems of linear equations) can be expressed by the following construct

```
!hpf$ independent
      do j=1,p
!  solve j-th system for coefficients f(1:q,j)
!  and store two last components of the result
!  in x(1:2,j)
!  .....
end do
```

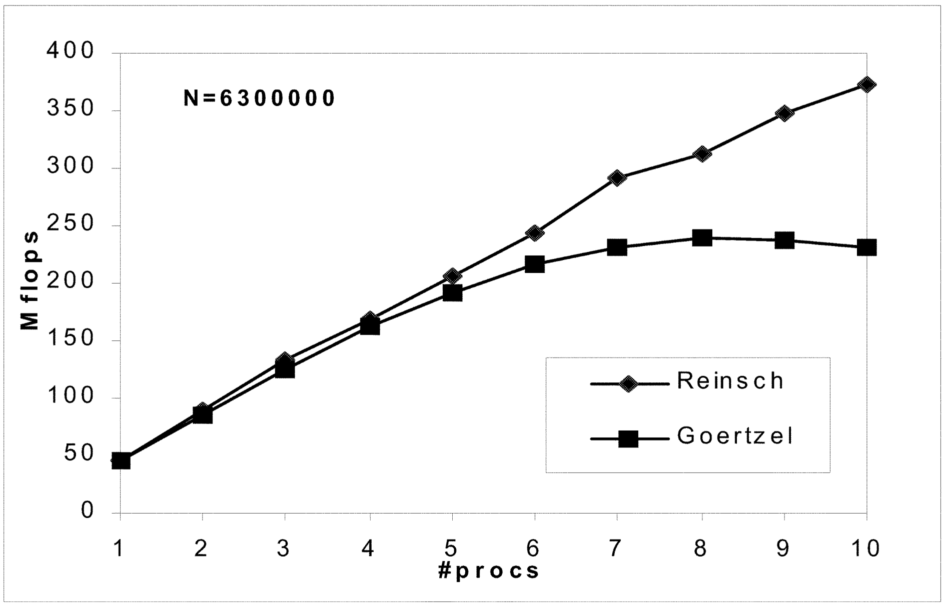

FIGURE 2: Performance of the parallel algorithms for $N = 6300000$

Further steps of the algorithms are typically sequential. Note that the constructions in which we implement (26) for Reinsch and (13) for Goertzel require communication between processors. However all necessary calls to communication subroutines are automatically generated by a HPF compiler.

We have tested our parallel programs on a cluster of ten PCs with Pentium II/400 processors running under Linux operating system with PVM3 message passing system [4]. As an HPF compiler, we have used ADAPTOR compilation system [1, 2]. Results of the experiments can be summarized as follows.

1. Both algorithms achieve better performance for greater problem sizes (see Figure 1). It is obvious, because in our algorithms the total amount of communication does not depend on the problem size and it is the same for both of them. Because of Reinsch performs more flops, thus it achieves better performance than Goertzel. Its efficiency is about 80.
2. For each problem size there exists an optimal number of processors (p_{opt}) for which the best performance is achieved. When we apply p processors, $p > p_{opt}$, the performance decreases (see Figures 2 and 3). For Reinsch, the value of p_{opt} is greater than for Goertzel.
3. We have not observed any significant loss of accuracy in comparison with corresponding sequential algorithms. Moreover, for Reinsch,

the accuracy is better for greater number of processors. Though it will be a subject of our further research.

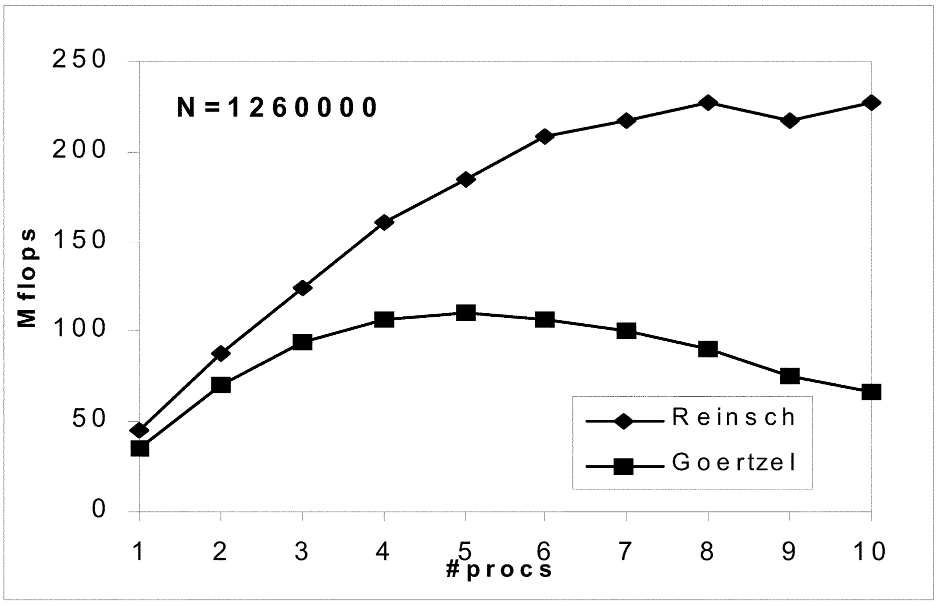

FIGURE 3: Performance of the parallel algorithms for $N = 1260000$

4. Conclusions. We have presented new efficient parallel versions of Goertzel and Reinsch algorithms for computing trigonometric sums. In the algorithms, we have applied the *divide–and–conquer* approach for solving problems in parallel. The additional improvements have been obtained by studying the special structure of linear recurrence systems formulated according to the sequential algorithms. We have shown how to reduce the number of subsystems to be solved in parallel. The new algorithms have been implemented in HPF and tested on a cluster of PCs with PVM3 message passing system. For a big problem size the efficiency of the parallel Reinsch algorithm is about eighty percent.

REFERENCES

[1] Brandes, T., R. Höver-Klier, *ADAPTOR User's Guide (Version 6.1)*, Technical documentation, GMD, 1998.

[2] Brandes, T., *ADAPTOR Programmers's Guide (Version 6.1)*, Technical documentation, GMD, 1998.

[3] Koelbel, C., D. Loveman, R. Shreiber, G. Steele and M. Zosel, *The High Performance Fortran Handbook*, MIT Press, Cambridge, 1994.

[4] Geist, A., A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam, *PVM: Parallel Virtual Machine. A Users's Guide and Tutorial for Networked Parallel Computing*, The MIT Press, Cambridge, 1994.

[5] Paprzycki, M., P. Stpiczyński, *Solving linear recurrence systems on a Cray Y-MP*, in: Lecture Notes in Computer Science 879 (J.Dongarra, J. Waśniewski, eds.), Springer–Verlag, Berlin, 1994, pp. 416–424.

[6] Paprzycki, M., P. Stpiczyński, *Solving linear recurrence systems on parallel computers*, Proceedings of the Mardi Gras '94 Conference, Baton Rouge, Feb. 10–12, 1994, Nova Science Publishers, New York, 1995, pp. 379–384.

[7] Paprzycki, M., P. Stpiczyński, *Parallel solution of linear recurrence systems*, Z. Angew. Math. Mech. **76** (1996), Suppl. 2, 5–8.

[8] Sameh, A.H., R.P. Brent, *Solving triangular systems on a parallel computer*, SIAM J. Numer. Anal. **14** (1977), 1101–1113.

[9] Stoer, J., R. Bulirsh, *Introduction to Numerical Analysis*, 2nd ed., Springer, New York, 1993.

[10] Stpiczyński, P., *Parallel algorithms for solving linear recurrence systems*, in: Lecture Notes in Computer Science 634 (L. Bougé et al., eds.), Springer–Verlag, Berlin, 1992, pp. 343–348.

[11] Stpiczyński, P., *Error analysis of two parallel algorithms for solving linear recurrence systems*, Parallel Comput. **19** (1993), 917–923.

[12] Stpiczyński, P., M. Paprzycki, *Parallel algorithms for finding trigonometric sums*, in: Parallel Processing for Scientific Computing (D.H. Bailey et al., eds.), SIAM, Philadelphia, 1995, pp. 291–292.

Department of Computer Science
Maria Curie-Skłodowska University
Pl. M. C.-Skłodowskiej 1
20-031 Lublin, Poland
e-mail: przem@hektor.umcs.lublin.pl

received June 13, 2001