# Parallel Performance of a Direct Elliptic Solver

Harry Hope* Marcin Paprzycki† Svetozara Petrova ‡

**Abstract**

In a recent paper [3] a parallel direct solver for the linear equations arising from elliptic partial differential equations has been proposed. The aim of this paper is to present the initial evaluation of the performance characteristics of the algorithm on a cluster of RS 6000 workstations, the SGI Power Challenge 8000 and 10000 shared memory computers and the SGI Origin 2000 dynamic shared memory computer.

**Key words:** elliptic equations, problems with sparse right-hand side, separation of variables, parallel performance

## 1 Introduction

Recently a new parallel algorithm for the solution of separable second order elliptic PDE's on rectangular domains has been presented by Petrova [3]. The method is based on earlier works of Vassilevski [5, 6]. The algorithm consists of odd-even block elimination technique combined with discrete separation of variables. The initial results indicated that the proposed solver has good numerical properties for 2D and 3D problems, [3, 5, 6, 7]. The aim of this paper is to present a more extensive study of its parallel performance characteristics.

The remainder of the paper is organized as follows. In Section 2 the method of discrete separation of variables is presented. Section 3 contains a brief summary of the Fast Algorithm for Separation of Variables (FASV) (for more details, see [3, 5]). Finally, Section 4 presents the results of experiments performed on a cluster of 8 IBM RS6000 workstations, the SGI Power Challenge 8000 and 10000 shared memory computers and an SGI Origin 2000 dynamic shared memory computer. We conclude with the description of the future research.

*Department of Mathematics and Computer Science, University of Texas of the Permian Basin, Odessa, TX 79762, USA, e-mail: hahiii@apex2000.net

†Department of Computer Science and Statistics, University of Southern Mississippi, Hattiesburg, MS 39406, USA, e-mail: marcin@usm.edu

‡Central Laboratory of Parallel Processing, Bulgarian Academy of Sciences, Acad. G.Bonchev Str., Block 25A, 1113 Sofia, Bulgaria, e-mail: zara@iscbg.acad.bg

## 2 Discrete Separation of Variables

Consider a separable second order elliptic equation of the form

$$-\sum_{s=1}^{d} \frac{\partial}{\partial x_s} a_s(x_s) \frac{\partial u}{\partial x_s} = f(x), \ x \in \Omega, \quad d = 2 \ \text{ or } 3$$

$$u_{|\partial\Omega} = 0.$$

Assume that $\Omega$ is a rectangle ($d = 2$), but the method described below can be generalized for 3D problems. Using, for example, the finite difference method to discretize the equation, we get the following linear system of equations

$$A\underline{x} = \underline{f}, \tag{2.1}$$

where

$$A = I_m \otimes T + B \otimes I_n.$$

$I_n$ is the identity $n \times n$ matrix and $\otimes$ is the *Kronecker product*. The matrices $T = (t_{ij})_{i,j=1}^{n}$ and $B = (b_{kl})_{k,l=1}^{m}$ are tridiagonal s.p.d. arising from the finite difference approximation of the one-dimensional operators $-\frac{\partial}{\partial x_s} a_s(x_s) \frac{\partial}{\partial x_s}(.)$, for $s = 1, 2$, respectively.

The solution $\underline{x}$ and the right-hand side $\underline{f}$ of (2.1) are composed using lexicographic ordering on horizontal lines. To describe the method of separation of variables we need also the vectors $\underline{x}'$ and $\underline{f}'$ reordered using vertical lexicographic ordering.

Consider now the following eigenvalue problem

$$B\underline{q}_k = \lambda_k \underline{q}_k, \tag{2.2}$$

where $\left\{\lambda_k, \underline{q}_k\right\}_{k=1}^{m}$ are the eigenpairs of $B_{m\times m}$. The matrix $B$ is assumed s.p.d. and hence the eigenvalues $\lambda_k > 0$ and the eigenvectors satisfy $\underline{q}_k^T \underline{q}_r = \delta_{kr}$ ($\delta_{kr}$ is the Kronecker symbol) for all $k, r = 1, 2, \ldots, m$.

Using the basis $\{\underline{q}_k\}_{k=1}^{m}$ the vectors $\underline{x}'_i$ and $\underline{f}'_i$ can be expanded as follows:

$$\underline{x}'_i = \sum_{k=1}^{m} \eta_{ki} \underline{q}_k, \qquad \underline{f}'_i = \sum_{k=1}^{m} \beta_{ki} \underline{q}_k, \quad i = 1, 2, \ldots, n, \tag{2.3}$$

where the Fourier coefficients of $\underline{f}'_i$ are computed by

$$\beta_{ki} = \underline{q}_k^T \underline{f}'_i. \tag{2.4}$$

Consider the column vectors $\underline{\eta}_k = (\eta_{ki})_{i=1}^{n}$ and $\underline{\beta}_k = (\beta_{ki})_{i=1}^{n}$, $k = 1, 2, \ldots, m$. Then substituting expressions (2.3) in (2.1) one gets the following system of equations for the discrete Fourier coefficients $\underline{\eta}_k$ of $\underline{x}'_i$, $i = 1, 2, \ldots, n$

$$(\lambda_k I + T)\underline{\eta}_k = \underline{\beta}_k, \quad k = 1, 2, \ldots, m. \tag{2.5}$$

Equations (2.5) represent $m$ systems of linear equations with matrices $n \times n$. They can be solved independently of each other.

We can now formulate the algorithm for the separation of variables (SV)

## Algorithm SV

(1) **determine** the eigenpairs $\left\{\lambda_k, \underline{q}_k\right\}_{k=1}^{m}$ of the tridiagonal matrix $B$ from (2.2);
(2) **compute** the Fourier coefficients $\beta_{ki}$ of $\underline{f}'_i$ using (2.4);
(3) **solve** $m$ $n \times n$ tridiagonal systems of equations of the form (2.5) to determine $\{\eta_{ki}\}$ – the Fourier coefficients of $\underline{x}'_i$, $i = 1, 2, \ldots, n$;
(4) **recover** the solution of our original system (2.1) on the basis of the Fourier coefficients $\{\eta_{ki}\}$ of $\underline{x}'_i$. There are two possibilities:

$$\begin{aligned} &\text{vertical recovering:} && \underline{x}'_i = \sum_{k=1}^{m} \eta_{ki} \underline{q}_k, \quad i = 1, 2 \ldots, n \\ &\text{horizontal recovering:} && \underline{x}_j = \sum_{k=1}^{m} q_{jk} \underline{\eta}_k, \quad j = 1, 2 \ldots, m. \end{aligned} \tag{2.6}$$

Let us assume that the systems of the form (2.1) have a sparse right-hand side (see for instance Banegas [1], Proskurowski [4] and Kuznetsov [2]). More precisely, assume that the right-hand side $\underline{f}$ has only $d \ll m$ nonzero block components

$$\underline{f}^T = \left[\underline{0}, \ldots, \underline{f}_{j_1}, \underline{0}, \ldots, \underline{f}_{j_d}, \underline{0}, \ldots, \underline{0}\right]^T, \quad \text{where } \underline{f}_{j_s} \in \mathcal{R}^n, \ s = 1, 2 \ldots, d.$$

Then for the reordered right-hand side each vector $\underline{f}'_i$ $(i = 1, 2, \ldots, n)$ has only $d$ nonzero scalar components $f_{ij_s}$, $s = 1, 2 \ldots, d$, i.e.

$$\underline{f}'^T_i = [0, \ldots, f_{ij_1}, 0, \ldots, f_{ij_d}, 0, \ldots, 0]^T, \quad i = 1, 2, \ldots, n.$$

Assume that only $r \ll m$ block components of the solution are needed. Denote by $\underline{x}_{j'_1}, \underline{x}_{j'_2}, \ldots, \underline{x}_{j'_r}$ the sought vectors.

To find the solution of a problem with a sparse right-hand side (SRHS) we apply the Algorithm SV for separation of variables

## Algorithm SRHS

(1) **compute** the Fourier coefficients $\beta_{ki}$ of $\underline{f}'_i$ from (2.4),

$$\beta_{ki} = \underline{q}_k^T \underline{f}'_i = \sum_{s=1}^{d} q_{kj_s} f_{ij_s}, \quad k = 1, 2, \ldots, m, \quad i = 1, 2, \ldots, n;$$

(2) **solve** systems of the form (2.5);
(3) **recover** the solution per lines using (2.6). We need only $\underline{x}_j = \sum_{k=1}^{m} q_{jk} \underline{\eta}_k$, for $j = j'_1, j'_2, \ldots, j'_r$.

# 3 Fast Algorithm for Separation of Variables

Consider now the algorithm FASV proposed by Vassilevski [5, 6] for solving problems of type (2.1). For simplicity we assume that $m = 2^l - 1$. The algorithm consists of two steps – forward and backward recurrence. For the forward step we need matrix $A$ in the following block form:

$$A = \begin{bmatrix} A^{(k,1)} & A_{12} & & & & 0 \\ A_{21} & T + b_{2^k 2^k} I_n & A_{23} & & & \\ & A_{32} & A^{(k,2)} & A_{34} & & \\ & & \ddots & \ddots & \ddots & \\ 0 & & & A_{2^{l-k+1}-1,2^{l-k+1}-2} & & A^{(k,2^{l-k})} \end{bmatrix}, \tag{3.1}$$

where

$$A^{(k,s)} = I_{2^k-1} \otimes T + B_k^{(s)} \otimes I_n, \quad s = 1, 2, \ldots, 2^{l-k},$$

i.e. each matrix $A^{(k,s)}$, $1 \le k \le l$, $1 \le s \le 2^{l-k}$ has $2^k - 1$ blocks of order $n$.

Above $B_k^{(s)}$ of order $2^k - 1$ is the principal submatrix of $B$ and hence, it is also a tridiagonal s.p.d. matrix of the form $B_k^{(s)} = (b_{ij})$, $i, j = 1, \ldots s_k + 2^k - 1$, where $s_k = (s-1)2^k$.

(1) **Forward step** of FASV

For $k = 1, 2, \ldots, l-1$ solve the problem:

$$A^{(k,s)} \underline{x}^{(k,s)} = \underline{f}^{(k,s)}, \quad s = 1, 2, \ldots, 2^{l-k}, \tag{3.2}$$

where

$$\underline{x}^{(k,s)} = \begin{bmatrix} \underline{x}_{s_k+1}^{(k)} \\ \underline{x}_{s_k+2}^{(k)} \\ \vdots \\ \underline{x}_{s_k+2^k-1}^{(k)} \end{bmatrix}$$

and $\underline{f}^{(k,s)}$ will be defined below.

After solving these problems and setting $\underline{x}_{s2^k}^{(k)} = 0$ for $s = 1, 2, \ldots, 2^{l-k} - 1$ we denote by $\underline{x}^{(k)}$ and $\underline{f}^{(k)}$ the following vectors

$$\underline{x}^{(k)} = \begin{bmatrix} \underline{x}^{(k,1)} \\ \underline{0} \\ \underline{x}^{(k,2)} \\ \underline{0} \\ \vdots \\ \underline{x}^{(k,2^{l-k})} \end{bmatrix} \begin{matrix} \}2^k - 1 \text{ blocks} \\ \}1 \text{ block} \\ \}2^k - 1 \text{ blocks} \\ \}1 \text{ block} \\ \\ \}2^k - 1 \text{ blocks} \end{matrix} \quad \text{and} \quad \underline{f}^{(k)} = \begin{bmatrix} \underline{f}^{(k,1)} \\ \underline{f}_{2^k} \\ \underline{f}^{(k,2)} \\ \vdots \\ \underline{f}^{(k,2^{l-k})} \end{bmatrix}. \tag{3.3}$$

Let the residual vector be the right-hand side for the next $k+1$st step

$$\underline{f}^{(k+1)} = \underline{f}^{(k)} - A\underline{x}^{(k)} = \begin{bmatrix} \underline{f}^{(k+1,1)} \\ \underline{f}_{2^{k+1}} \\ \underline{f}^{(k+1,2)} \\ \vdots \\ \underline{f}^{(k+1,2^{l-k-1})} \end{bmatrix}, \tag{3.4}$$

where

$$\underline{f}^{(k+1,s')} = \begin{bmatrix} \underline{0} \\ \underline{f}^{(k+1)}_{s'.2^{k+1}} \\ \underline{0} \end{bmatrix} \begin{matrix} \}2^k-1 \text{ blocks} \\ \}1 \text{ block} \\ \}2^k-1 \text{ blocks} \end{matrix}, \quad s' = 1, 2, \ldots, 2^{l-k-1}.$$

From (3.4) and $s = (2s'-1)$ we have

$$\begin{aligned} \underline{f}^{(k+1)}_{s'.2^{k+1}} &= \underline{f}^{(k)}_{s.2^k} - A_{2s,2s-1}\,\underline{x}^{(k,s)} - A_{2s,2s+1}\,\underline{x}^{(k,s+1)} \\ &= \underline{f}^{(k)}_{s.2^k} - b_{s.2^k,s.2^k-1}\,\underline{x}^{(k,s)}_{2^k-1} - b_{s.2^k,s.2^k+1}\,\underline{x}^{(k,s+1)}_{1}. \end{aligned} \tag{3.5}$$

The new right-hand side $\underline{f}^{(k+1,s')}$ has only one nonzero block component and hence, by induction, the right-hand sides $\underline{f}^{(k,s)}$ have the following sparsity pattern

$$\underline{f}^{(k,s)} = \begin{bmatrix} \underline{0} \\ * \\ \underline{0} \end{bmatrix} \begin{matrix} \}2^{k-1}-1 \text{ blocks} \\ \}1 \text{ block} \\ \}2^{k-1}-1 \text{ blocks} \end{matrix}, \quad s = 1, 2, \ldots, 2^{l-k}.$$

Therefore, the problems (3.2) have a sparse right-hand side. The matrices $A^{(k,s)}$ allow a separation of variables as submatrices of $A$ and hence, Algorithm SV from Section 2 can be applied with $d = 1$ (the number of nonzero block components of the right-hand side), and $r = 3$ (the number of the sought block components of the solution: $\underline{x}^{(k,s)}_1$, $\underline{x}^{(k,s)}_{2^{k-1}}$ and $\underline{x}^{(k,s)}_{2^k-1}$).

(2) **Backward step** of FASV

For $k = l, l-1, \ldots, 1$, our purpose is to determine $\underline{x}_{(2s-1)2^{k-1}}$ for $s = 1, 2, \ldots, 2^{l-k}$. First, when $k = l$, we solve

$$A\underline{x}^{(l)} = \underline{f}^{(l)}, \tag{3.6}$$

where $A^{(l,s)} \equiv A$, $\underline{x}^{(l)} = \underline{x}^{(l,1)}$ and $\underline{f}^{(l)} = \underline{f}^{(l,1)}$. The right-hand side $\underline{f}^{(l)}$ is found at the last step of the forward recurrence and from (3.5) it has a sparse right-hand side. The problem (3.6) is solved incompletely finding only one block component $\underline{x}^{(l)}_{2^{l-1}}$.

We have also $\underline{x}_{2^{l-1}} = \underline{x}^{(l)}_{2^{l-1}}$ which corresponds to the midblock component of the solution $\underline{x}$ of (2.1). The remaining block components of $\underline{x}$ are recovered by induction as follows:

Starting with $k = l$ we have $\underline{x}_{2^{l-1}} = \underline{x}^{(l)}_{2^{l-1}}$. Assume that for some given $k$, $1 \leq k \leq l-1$, we have found the vectors $\underline{x}_{s.2^k}$ for $s = 1, 2, \ldots, 2^{l-k} - 1$ in the previous steps $k+1, \ldots, l$.

Then at the $k$th step we can find $\underline{x}_{s.2^{k-1}}$ for $s = 1, 2, \ldots, 2^{l-k+1} - 1$. More precisely, by construction we have $\underline{f}^{(k'+1)} = \underline{f}^{(k')} - A\underline{x}^{(k')}$ and after summation of both sides of these equalities for $k' = k, k+1, \ldots, l$, we get

$$\underline{f}^{(k)} = A\left(\sum_{k'=k}^{l} \underline{x}^{(k')}\right). \tag{3.7}$$

Let

$$\underline{y} = \sum_{k'=k}^{l} \underline{x}^{(k')} \tag{3.8}$$

and using $A$ from (3.1) we have the following equivalent form for the system (3.7)

$$\begin{bmatrix} A^{(k,1)} & A_{12} & & & & 0 \\ A_{21} & T + b_{2^k 2^k} I_n & A_{23} & & & \\ & A_{32} & A^{(k,2)} & A_{34} & & \\ & & \ddots & \ddots & \ddots & \\ 0 & & & A_{2^{l-k+1}-1, 2^{l-k+1}-2} & & A^{(k,2^{l-k})} \end{bmatrix} \begin{bmatrix} \underline{y}^{(k,1)} \\ \underline{y}_{2^k} \\ \underline{y}^{(k,2)} \\ \vdots \\ \underline{y}^{(k,2^{l-k})} \end{bmatrix} = \begin{bmatrix} \underline{f}^{(k,1)} \\ \underline{f}_{2^k} \\ \underline{f}^{(k,2)} \\ \vdots \\ \underline{f}^{(k,2^{l-k})} \end{bmatrix}$$

where each block $\underline{y}^{(k,s)}$, $s = 1, 2, \ldots, 2^{l-k}$ has $2^k - 1$ blocks of order $n$. From this system we obtain the following equation for $\underline{y}^{(k,s)}$ :

$$A_{2s-1,2s-2}\underline{y}_{(s-1).2^k} + A^{(k,s)}\underline{y}^{(k,s)} + A_{2s-1,2s}\underline{y}_{s.2^k} = \underline{f}^{(k,s)}. \tag{3.9}$$

Hence, when $s_k = (s-1)2^k$:

$$A^{(k,s)}\underline{y}^{(k,s)} = \underline{f}^{(k,s)} - A_{2s-1,2s-2}\underline{y}_{s_k} - A_{2s-1,2s}\underline{y}_{s.2^k}. \tag{3.10}$$

Using (3.8) we have $\underline{y}_{s_k} = \underline{x}_{s_k}$ and $\underline{y}_{s.2^k} = \underline{x}_{s.2^k}$. Recall that from the induction described above, the vectors $\underline{x}_{s_k} = \underline{x}_{(s-1).2^k}$ and $\underline{x}_{s.2^k}$ are already found. Thus, from (3.10) we get the following system

$$A^{(k,s)}\underline{y}^{(k,s)} = \begin{bmatrix} -b_{s_k+1,s_k}\underline{x}_{s_k} \\ \underline{0} \\ \underline{f}^{(k,s)}_{2^{k-1}} \\ \underline{0} \\ -b_{s.2^k-1,s.2^k}\underline{x}_{s.2^k} \end{bmatrix} \begin{matrix} \}1 \text{ st block} \\ \\ \}2^{k-1} \text{ th block} \\ \\ \}2^k - 1 \text{ st block} \end{matrix} . \tag{3.11}$$

The nonzero block components of the right-hand side of (3.11) can be found using the solution computed at the $k+1$st step. It is a problem with a sparse right-hand side and

only one ($r = 1$) block component of the solution, namely $\underline{y}^{(k,s)}_{2^{k-1}}$ is needed. By (3.8) one finds

$$\underline{y}^{(k,s)}_{2^{k-1}} = \underline{y}_{(2s-1).2^{k-1}} = \sum_{k'=k}^{l} \underline{x}^{(k')}_{(2s-1).2^{k-1}} = \underline{x}_{(2s-1).2^{k-1}}.$$

Therefore, at the $k$th step from the backward recurrence ($k = l, l-1, \ldots, 1$) we can determine $\underline{x}_{(2s-1)2^{k-1}}$, $s = 1, 2, \ldots, 2^{l-k}$.

In a more complete form the Fast Algorithm for Separation of Variables (FASV) can be described as follows

## Algorithm FASV

(1) **Forward step**

**Set** $\underline{f}^{(1)} = \underline{f}$;

**For** $k = 1$ to $l - 1$

**for** $s = 1$ to $2^{l-k}$ **solve** incompletely,

$$A^{(k,s)}\underline{x}^{(k,s)} = \begin{bmatrix} \underline{0} \\ \underline{f}^{(k,s)}_{2^{k-1}} \\ \underline{0} \end{bmatrix}; \tag{3.12}$$

finding only

$$\underline{x}^{(k,s)}_1, \ \underline{x}^{(k,s)}_{2^{k-1}} \ \text{ and } \ \underline{x}^{(k,s)}_{2^k-1};$$

**end** { loop on $s$ };

**for** $s' = 1$ to $2^{l-k-1} - 1$ and $s = (2s' - 1)$ **compute**

$$\underline{f}^{(k+1)}_{s'.2^{k+1}} = \underline{f}^{(k)}_{s.2^k} - b_{s.2^k, s.2^k-1}\underline{x}^{(k,s)}_{2^k-1} - b_{s.2^k, s.2^k+1}\underline{x}^{(k,s+1)}_1; \tag{3.13}$$

**end** { loop on $s'$ };

**end** { loop on $k$ };

(2) **Backward step**

**Solve** incompletely

$$A\underline{x}^{(l)} = \begin{bmatrix} \underline{0} \\ \underline{f}^{(l)}_{2^{l-1}} \\ \underline{0} \end{bmatrix}; \tag{3.14}$$

only for $\underline{x}^{(l)}_{2^{l-1}} = \underline{x}_{2^{l-1}}$.

**For** $k = l - 1$ down to 1

**for** $s = 1$ to $2^{l-k}$ **solve** incompletely

$$A^{(k,s)}\underline{y} = \begin{bmatrix} -b_{s_k+1, s_k}\underline{x}_{s_k} \\ \underline{0} \\ -b_{s.2^k-1, s.2^k}\underline{x}_{s.2^k} \end{bmatrix}, \quad \text{where} \quad s_k = (s-1)2^k \tag{3.15}$$

| P | RS6K | | PC8K | | PC10K | | O2K | |
|---|---|---|---|---|---|---|---|---|
| | T | S | T | S | T | S | T | S |
| 1 | 91.32 | 1 | 4.19 | 1 | 3.67 | 1 | 3.15 | 1 |
| 2 | 50.18 | 1.81 | 4.36 | 0.96 | 2.49 | 1.47 | 2.31 | 1.36 |
| 4 | 24.61 | 3.71 | 2.74 | 1.52 | 1.53 | 2.39 | 2.23 | 1.41 |
| 8 | 13.65 | 6.69 | 2.26 | 1.85 | 2.14 | 1.71 | 3.06 | 1.02 |
| 12 | | | 1.79 | 2.34 | 1.72 | 2.13 | 4.25 | 0.74 |

Table 1: Performance comparison, l=10

to determine only $\underline{y}_{2^{k-1}}$.
Then **set**

$$\underline{x}_{(2s-1)2^{k-1}} = \underline{y}_{2^{k-1}} + \underline{x}^{(k,s)}_{2^{k-1}}; \tag{3.16}$$

**end** { loop on $s$ };
**end** { loop on $k$ }; □

# 4 Experimental results

The proposed algorithm has been implemented in FORTRAN 77 using the PVM message passing environment to establish its parallel performance characteristics. We have experimented on an 8-computer network of IBM RS6000 workstations (RS6K), on two 16-processor Silicon Graphics Power Challenge shared memory computers (based on MIPS 8000 (PC8K) and MIPS 10000 (PC10K) processors) and, finally, on the Silicon Graphics Origin 2000 (O2K) dynamic shared memory computer (also based on the MIPS 10000 processors). Execution time has been measured using the *mclock* utility. In addition, on each machine, at least one more timer-utility has been used to confirm the results (for instance on the SGI machines the *getrusage* utility has been used). All experiments have been performed either in a benchmarking environment, or on lightly loaded systems. In the latter case the best result of multiple runs is reported.

In Table 1 we present the execution times in seconds (T) and speed-up (S) for all four machines for $P = 1, 2, 4, 8$ and 12 processors for $l = 10$ (2D grid of size $n = 2^l = 1024$).

The results illustrate the effect of the computation to communication ratio. Here, the best speed-up is achieved on the workstation cluster. However, the solution time on 8 workstations is about three times slower than a one processor solution on the SGI computers. It is clear that in the case of the workstations the relative slowness of the processors hides the relative slowness of the interconnection network. The situation is exactly the opposite in the case of the shared memory machines. Even if the network is relatively fast, the speed of the processors reduces the gains from parallelization. It should be also observed that on the Origin 2000 the performance decreases when the number of processors increases above $P = 8$. This can be attributed to the fact that the Origin consists of tightly coupled clusters of 8 processors. Thus, for $P > 8$ more than one cluster

is utilized and the communication time increases considerably. For the SGI computers the problem of size $l = 10$ is too small for efficient parallelization. We were able to fit problems of size $l = 11$ and 12 into the memory of the shared memory machines. The speed-up achieved for all three problem sizes for $P = 1, 2, \ldots, 13$ processors is presented in Figure 1.

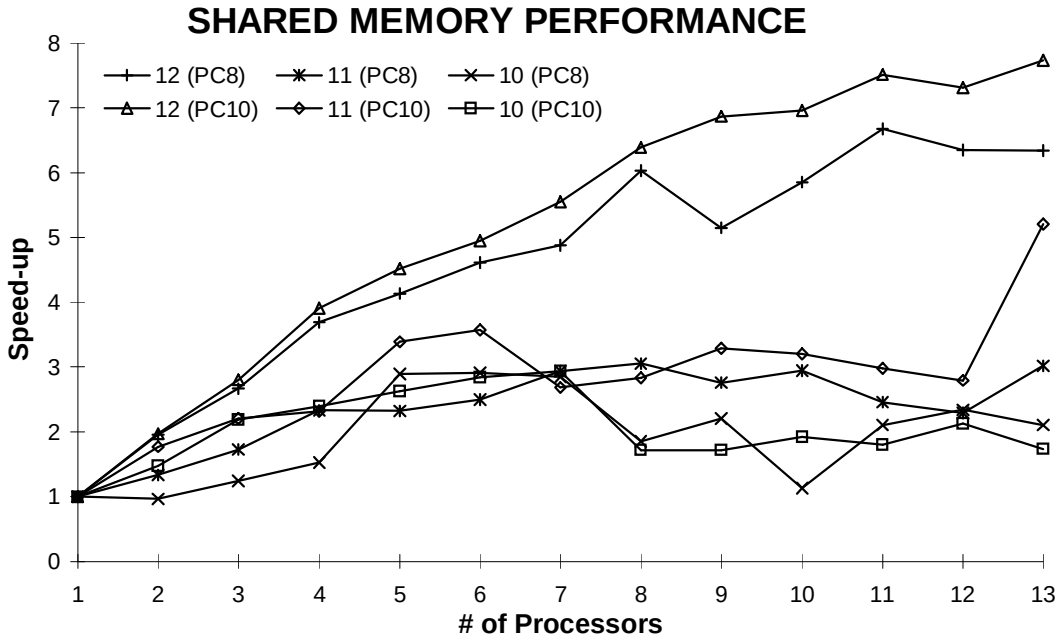


Figure 1: Performance comparison between the two PC machines.

It can be observed that only the largest problem is large enough for efficient parallelization. Efficiency of about 50% was achieved on the PC8K and about 60% on the PC10K. Interestingly, not only PC10K is faster than PC8K (on one processor, a problem of size $l = 12$ is solved about 1.5 times faster) but due to the better hardware efficiency this ratio increases with the number of processors (reaching about 1.8 on 13 processors). We have tried to further increase the number of processors used, but (as can be observed in Figure 1, where the speed-up curves become flat) this did not lead to further performance increases.

# 5 Concluding remarks

In this paper the experimental results illustrating the parallel performance characteristics of the fast elliptic solver have been presented. The code turned out to be rather efficient (even though message passing environment was used to implement it on shared memory machines). We have also found that memory capacity is the primary factor restricting the size of problems solved. In the near future we plan to extend our research in the following directions:

- implementation of the code using the shared memory approach and a performance comparison between the two approaches on shared memory machines,
- comparison of the performance of the PVM implementation and the shared memory implementation on dynamic shared memory machines,
- extension of the proposed approach to the parallel three dimensional solvers.

# 6 Acknowledgements

The computer time grants from the ULB, Brussels, Belgium and the NCSA in Urbana are kindly acknowledged. The research has been initiated thanks to the COBASE grant from the National Research Council. The research of the third author was partially supported by the Bulgarian Ministry of Education, Science and Technology under Grants MM #415/94 and MU-MM #01/96. We are grateful to Katarzyna Paprzycka for help with English.

# References

[1] A. Banegas, Fast Poisson solvers for problems with sparsity, *Math.Comp.*, **32**(1978), 441-446.

[2] Y. Kuznetsov, Block relaxation methods in subspaces, their optimization and application, *Soviet.J.Numer.Anal. and Math. Modelling*, **4**(1989), 433-452.

[3] S. Petrova, Parallel implementation of fast elliptic solver, *Parallel Computing*, 1997 (to appear).

[4] W. Proskurowski, Numerical solution of Helmholtz equation by implicit capacitance matrix method, *ACM Trans. Math. Software*, **5**(1979), 36-49.

[5] P. Vassilevski, Fast algorithm for solving a linear algebraic problem with separation of variables, *Comptes Rendus de l'Academie Bulgare des Sciences*, **37**(1984), No.3, 305-308.

[6] P. Vassilevski, Fast algorithm for solving discrete Poisson equation in a rectangle, *Comptes Rendus de l'Academie Bulgare des Sciences*, **38**(1985), No.10, 1311-1314.

[7] P. Vassilevski and S. Petrova, A note on construction of preconditioners in solving 3D elliptic problems by substructuring, *Comptes Rendus de l'Academie Bulgare des Sciences*, **41**(1988), No.7, 33-36.