

Sequential and Parallel Algorithms for the Inverse Toeplitz Singular Value Problem

Pedro Alonso
 Departamento de Sistemas
 Informáticos y Computación
 Universidad Politécnica de Valencia
 Camino de Vera s/n,
 46022 Valencia, España
 Email: palonso@dsic.upv.es

Georgina Flores-Becerra
 Departamento de Sistemas
 y Computación
 Instituto Tecnológico de Puebla
 Av. Tecnológico 420,
 72220 Puebla, México
 Email: gflores@dsic.upv.es

Antonio M. Vidal
 Departamento de Sistemas
 Informáticos y Computación
 Universidad Politécnica de Valencia
 Camino de Vera s/n,
 46022 Valencia, España
 Email: avidal@dsic.upv.es

Abstract—When the Inverse Additive Singular Value Problem (IASVP) involves Toeplitz-type matrices it is possible to exploit this special structure to reduce the execution time. In this paper, we present two iterative local and global convergent algorithms (MIIT and LPT) to solve efficiently the IASVP when the matrix is Toeplitz (IASVPT). As it will be shown, it can be achieved an asymptotic complexity one order of magnitude less than those algorithms that do not exploit the Toeplitz-like structure. Furthermore, we have implemented a parallel version of these both algorithms, called PMIIT and PLPT, respectively, that highly reduce the execution time of the sequential algorithm at sight of the experiments.

Keywords— Parallel programming, Inverse Singular Value Problem, Toeplitz matrices, Newton type methods, Least squares problem

1. Introduction

Inverse problems appear in different applications such as the determination of mass distributions, orbital mechanics, irrigation theory, computed tomography, circuit theory, among others [1, 2, 3]. In particular, the Inverse Additive Singular Value Problem (IASVP) is a problem of parameter identification. IASVP consists of the reconstruction of a matrix with prescribed singular values and with certain structure.

Given a set of matrices $A_0, A_1, \dots, A_n \in \mathbb{R}^{m \times n}$ ($m \geq n$) and a set of real numbers $S^* = \{S_1^*, S_2^*, \dots, S_n^*\}$, where $S_1^* \geq S_2^* \geq \dots \geq S_n^*$, we search for a vector $c = [c_1, c_2, \dots, c_n]^T \in \mathbb{R}^n$, such that S^* are the singular values of matrix

$$A(c) = A_0 + c_1 A_1 + \dots + c_n A_n. \quad (1)$$

There exist Newton-type algorithms to solve the IASVP [4, 5] like BF, MI and MIII; or the LP algorithm which is a distance minimization method. All of these iterative algorithms compute a succession of vectors $c^{(0)}, c^{(1)}, \dots, c^{(k)}$ that approximates c^* , that is, the solution of the IASVP. The MI, MIII and LP algorithms have been parallelized under the distributed memory model [6, 7].

In this paper we consider the particular case of the IASVP when matrices A_i ($i = 0, 1, \dots, n$) are Toeplitz. Although this case could be solved by using the mentioned general algorithms, it is possible to improve the performance if specific algorithms which exploit the matrix structure are designed. We present both sequential and parallel algorithms to solve the IASVPT efficiently following this idea, specially to tackle large size problems.

In the following section the IASVPT is introduced. The sequential algorithms are shown in Section 3. Section 4 shows the parallelization of these sequential algorithms. The space and time complexities of the algorithms are analyzed, both theoretically and experimentally, making a comparison with existing algorithms to solve the IASVP. The paper concludes with a conclusion section.

All algorithms have been implemented using serial (BLAS [8], LAPACK [9]), and parallel (PBLAS [10] and ScaLAPACK [11]) portable standard packages of linear algebra and using BLACS [12] and MPI [13] as communication libraries. The numerical experiments have been executed on a cluster of sixteen PCs, each one with two-processors Intel Xeon, working at 2 GHz, with 1 GByte of RAM, and connected through a SCI network.

2. The IASVPT

In the IASVPT, given S^* , the problem is to build a Toeplitz matrix $T \in \mathbb{R}^{n \times n}$,

$$T = [t_{i-j}]_{i,j=1,n} = \begin{bmatrix} t_0 & t_{-1} & t_{-2} & \dots & t_{-(n-1)} \\ t_1 & t_0 & t_{-1} & \dots & t_{-(n-2)} \\ t_2 & t_1 & t_0 & \dots & t_{-(n-3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{(n-1)} & t_{(n-2)} & t_{(n-3)} & \dots & t_0 \end{bmatrix},$$

with singular values S^* , what means to find the values of $2n - 1$ unknowns, the components of T , such that the

singular value decomposition of T is $U\Sigma^*V^T$, where $\Sigma^* = \text{diag}(S^*)$.

Defining

$$\tilde{t}_i = \frac{t_i}{t_0},$$

for $i = -(n-1), -(n-2), \dots, 1, 2, \dots, n-2, n-1$; and

$$\tilde{S}_i^* = \frac{S_i^*}{t_0},$$

for $i = 1, 2, \dots, n$, the IASVPT can be expressed as finding $\tilde{T} = U\tilde{\Sigma}^*V^T$, where

$$\tilde{T}_{ij} = \begin{cases} \tilde{t}_{i-j} & i \neq j \\ 1 & i = j \end{cases} \quad i, j = 1, 2, \dots, n;$$

and \tilde{T} can be written as

$$\tilde{T} = I + \tilde{t}_1 G + \tilde{t}_2 G^2 + \dots + \tilde{t}_{n-1} G^{n-1} + \tilde{t}_{-1} G^T + \tilde{t}_{-2} (G^T)^2 + \dots + \tilde{t}_{-(n-1)} (G^T)^{n-1}, \quad (2)$$

where G is the following one position down shift matrix

$$G = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix}.$$

In (2) there are $2n-2$ parameters \tilde{t}_i ($i = -(n-1), -(n-2), \dots, 1, 2, \dots, n-2, n-1$) to be computed and $2n-1$ $(n \times n)$ -Toeplitz matrices. In order to express the IASVPT as an IASVP with possibly unique solution, we can take arbitrary values for $n-2$ parameters, thus achieving an IASVP with n unknowns to be computed to assign n prescribed singular values.

Assigning the values a_i ($i = 1, 2, \dots, n-2$) to the $n-2$ parameters \tilde{t}_i , equation (2) can be expressed as

$$\tilde{T} = \tilde{T}_0 + \tilde{t}_1 G + \tilde{t}_2 G^2 + \dots + \tilde{t}_p G^p + \tilde{t}_{-1} G^T + \tilde{t}_{-2} (G^T)^2 + \dots + \tilde{t}_{-q} (G^T)^q, \quad (3)$$

where $p+q = n$ and

$$\tilde{T}_0 = I + a_1 G^{p+1} + \dots + a_{n-p-1} G^{n-1} + a_{n-p} (G^T)^{q+1} + \dots + a_{n-2} (G^T)^{n-1}.$$

Now, in (3) there are n parameters \tilde{t}_i and $n+1$ data $(n \times n)$ -Toeplitz matrices represented through G . Therefore, equation (3) represents an IASVP with

$$\begin{aligned} c &= [c_1, c_2, \dots, c_p, c_{p+1}, \dots, c_n]^T = \\ &= [\tilde{t}_1, \tilde{t}_2, \dots, \tilde{t}_p, \tilde{t}_{-1}, \dots, \tilde{t}_{-q}]^T, \end{aligned}$$

and

$$\begin{aligned} A_0 &= \tilde{T}_0, \quad A_1 = G, \quad \dots, \quad A_p = G^p, \\ A_{p+1} &= G^T, \quad A_{p+2} = (G^T)^2, \quad \dots, \quad A_n = (G^T)^q. \end{aligned}$$

In this way MIII and LP algorithms to solve the IASVP can also be used to solve the IASVPT.

3. Efficient Sequential Solution for the IASVPT

This section begins with a brief introduction of MIII and LP algorithms to solve the IASVP.

Let us define $\Gamma(S^*)$, the set of matrices in $\mathfrak{R}^{m \times n}$, ($m \geq n$), which can be written in the form $U\Sigma^*V^T$, where $U \in \mathfrak{R}^{m \times n}$ and $V \in \mathfrak{R}^{n \times n}$ are orthogonal matrices; and let $\Lambda(c)$ be the set of matrices that can be expressed as in (1).

MIII is a method developed by Chu [4], which finds the intersection of $\Gamma(S^*)$ and $\Lambda(c)$, using an iterative Newton-type method (Algorithm 1). In iteration k , given $X^{(k)} \in \Gamma(S^*)$, there exist matrices $U^{(k)}$ and $V^{(k)}$ such that $X^{(k)} = U^{(k)}\Sigma^*V^{(k)T}$. The tangent vector to $\Gamma(S^*)$ which starts from the point $X^{(k)}$ and crosses $A(c^{(k+1)})$, can be expressed as

$$X^{(k)} + X^{(k)}L^{(k)} - H^{(k)}X^{(k)} = A(c^{(k+1)}), \quad (4)$$

where $L^{(k)} \in \mathfrak{R}^{n \times n}$ and $H^{(k)} \in \mathfrak{R}^{m \times m}$ are skew-symmetric matrices. Because $X^{(k)} = U^{(k)}\Sigma^*V^{(k)T}$, equation (4) can be expressed as

$$\Sigma^* + \Sigma^*\tilde{L}^{(k)} - \tilde{H}^{(k)}\Sigma^* = W^{(k)}, \quad (5)$$

where $\tilde{L}^{(k)} = V^{(k)T}L^{(k)}V^{(k)}$, $\tilde{H}^{(k)} = U^{(k)T}H^{(k)}U^{(k)}$, $W^{(k)} = U^{(k)T}A(c^{(k+1)})V^{(k)}$. Equating the diagonal elements of (5), we obtain the linear system $J^{(k)}c^{(k+1)} = b^{(k)}$ that allows to compute $c^{(k+1)}$ ($A(c^{(k+1)})$ and $W^{(k)}$), where

$$J^{(k)} = [u_i^{(k)T} A_j v_i^{(k)}]_{i,j=1,2,\dots,n}, \quad (6)$$

and

$$b^{(k)} = S^* - [u_i^{(k)T} A_0 v_i^{(k)}]_{i=1,2,\dots,n}. \quad (7)$$

Equating the off-diagonal elements of (5), we calculate $\tilde{H}^{(k)}$ and $\tilde{L}^{(k)}$ as

$$\begin{aligned} \tilde{H}_{ij}^{(k)} = -\tilde{H}_{ji}^{(k)} &= -\frac{W_{ij}^{(k)}}{S_j^{*2}}, \\ i &= n+1, \dots, m; \quad j = 1, \dots, n; \end{aligned} \quad (8)$$

$$\begin{aligned} \tilde{H}_{ij}^{(k)} = -\tilde{H}_{ji}^{(k)} &= \frac{S_i^* W_{ji}^{(k)} + S_j^* W_{ij}^{(k)}}{S_i^{*2} - S_j^{*2}}, \\ i &= 1, \dots, n; \quad j = i+1, \dots, n; \end{aligned} \quad (9)$$

$$\begin{aligned} \tilde{L}_{ij}^{(k)} = -\tilde{L}_{ji}^{(k)} &= \frac{S_i^* W_{ij}^{(k)} + S_j^* W_{ji}^{(k)}}{S_i^{*2} - S_j^{*2}}, \\ i &= 1, \dots, n; \quad j = i+1, \dots, n. \end{aligned} \quad (10)$$

Using $c^{(k+1)}$ to compute $X^{(k+1)}$, e.g. $U^{(k+1)}$ and $V^{(k+1)}$, $A(c^{(k+1)}) \in \Lambda(c)$ must be lifted to a point in $\Gamma(S^*)$. Then $X^{(k+1)}$ is defined as

$$X^{(k+1)} = U^{(k+1)}\Sigma^*V^{(k+1)T},$$

where $U^{(k+1)}$ and $V^{(k+1)}$ are orthogonal matrices which can be approximated by

$$U^{(k+1)} \approx U^{(k)} R,$$

and

$$V^{(k+1)} \approx V^{(k)} T,$$

being R and T the Cayley transforms:

$$R = \left(I + \frac{1}{2} H^{(k)} \right) \left(I - \frac{1}{2} H^{(k)} \right)^{-1},$$

and

$$T = \left(I + \frac{1}{2} L^{(k)} \right) \left(I - \frac{1}{2} L^{(k)} \right)^{-1}.$$

Therefore, $U^{(k+1)}$ and $V^{(k+1)}$ are approximated by solving the linear systems

$$\left(I + \frac{H^{(k)}}{2} \right) U^{(k+1)} = \left(I - \frac{H^{(k)}}{2} \right) U^{(k)}, \quad (11)$$

and

$$\left(I + \frac{L^{(k)}}{2} \right) V^{(k+1)} = \left(I - \frac{L^{(k)}}{2} \right) V^{(k)}. \quad (12)$$

MIII converges quadratically to the solution if $c^{(0)}$ is close enough to c^* [14]. See [4] for details.

Algorithm 1 MIII

1. Compute A in accordance with (1)
 2. Compute $[U, \Sigma, V] = svd(A)$
 3. For $k = 0, 1, \dots$, While $\|U^T A V - \Sigma^*\|_F > tol$
 - 3.1. Compute J, b in accordance with (6), (7)
 - 3.2. Solve $Jc = b$
 - 3.3. Compute A in accordance with (1)
 - 3.4. Compute \tilde{H}, \tilde{L} in accordance with (8)–(10)
 - 3.5. Approximate U, V solving (11), (12)
- End For
-

The goal of LP [7] is to find the intersection of $\Gamma(S^*)$ and $\Lambda(c)$ sets, using distance minimization techniques (Algorithm 2). The distance between two matrices P and Q is defined as $d(P, Q) = \|P - Q\|_F$. LP is an adaptation of the Lift&Project algorithm proposed by Chu in [15].

In iteration k , given $c^{(k)}$ (given $A(c^{(k)}) \in \Lambda(c)$) the problem is to find $X^{(k)} \in \Gamma(S^*)$ such that

$$d(A(c^{(k)}), X^{(k)}) = d(A(c^{(k)}), \Gamma(S^*)).$$

This is achieved by computing the singular value decomposition of $A(c^{(k)})$, $U^{(k)} \Sigma^{(k)} V^{(k)T}$, and computing $X^{(k)}$ as [16, 17]

$$X^{(k)} = U^{(k)} \Sigma^* V^{(k)T}.$$

To compute $c^{(k+1)}$ from $X^{(k)}$, such that

$$d(X^{(k)}, A(c^{(k+1)})) = d(X^{(k)}, \Lambda(c)),$$

the nonlinear least squares problem

$$\min_{c^{(k+1)}} \|A(c^{(k+1)}) - U^{(k)} \Sigma^* V^{(k)T}\|_F^2 = \min_{c^{(k+1)}} F^{(k+1)}$$

is solved by equating the gradient of $F^{(k+1)}$ to zero and solving the resulting linear system $A_{tr} c^{(k+1)} = b_{tr}^{(k)}$, where

$$A_{tr} = [tr(A_i^T A_r)]_{r,i=1,2,\dots,n}, \quad (13)$$

and

$$b_{tr}^{(k)} = [tr(A_r^T (X^{(k)} - A_0^T))]_{r=1,2,\dots,n}. \quad (14)$$

LP algorithm converges (analog to [15]) to a stationary point in the sense that

$$\|A(c^{(k+1)}) - X^{(k+1)}\|_F \leq \|A(c^{(k)}) - X^{(k)}\|_F.$$

Algorithm 2 LP

1. Compute A_{tr} in accordance with (13)
 2. For $k = 1, 2, \dots$
 - 2.1. Compute A in accordance with (1)
 - 2.2. Compute $[U, \Sigma, V] = svd(A)$
 - 2.3. Compute b_{tr} in accordance with (14)
 - 2.4. $cOld = c$
 - 2.5. Solve $A_{tr} c = b_{tr}$
- Until $\|c - cOld\|_2 < tol$
-

The IASVP can be adapted to take advantage of the structure of Toeplitz matrices of the IASVPT. The IASVPT needs to storage only $n - 2$ double precision scalars (\tilde{T}_0), a requirement drastically smaller than the $n^2(n + 1)$ double precision scalars to storage A_i , for $i = 0, 1, \dots, n$, in the case of the IASVP.

Moreover, the operations of MIII and LP that can be improved using the structures of the IASVPT matrices are the computation of $A(c)$, J , A_{tr} matrices; b_{tr} vector; and the solution of $A_{tr} c = b_{tr}$. The computation of $A(c)$ (1) consists of assign c_i to some component of $A(c)$ without floating point operations, then $A(c)$ is expressed as

$$\begin{bmatrix} 1 & c_{p+1} & \dots & c_n & a_{n-p} & \dots & a_{n-2} \\ c_1 & 1 & \dots & c_{n-1} & c_n & \dots & a_{n-3} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ c_p & c_{p-1} & \dots & 1 & c_{p+1} & \dots & c_n \\ a_1 & c_p & \dots & c_1 & 1 & \dots & c_{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{n-p-1} & a_{n-p-2} & \dots & c_p & c_{p-1} & \dots & 1 \end{bmatrix}.$$

The product of some \tilde{T}_i ($i = 1, 2, \dots, n$) by any n -vector consists of shifting the vector elements in accordance with the position of value 1 into the \tilde{T}_i , whereby the computation of J (6) is reduced to dot products:

$$\begin{aligned} J_{ij} &= u_i^T A_j v_i = \\ &= \text{dot_product}(u_i^T, \\ &\quad \text{shifts_vector_elements}(A_j, v_i)). \end{aligned}$$

The computation of A_{tr} (13) does not involve floating point operations because its definition only involves matrix G , so A_{tr} is the following diagonal matrix

$$A_{tr_{ij}} = \begin{cases} n - i & i = j \\ 0 & i \neq j \end{cases},$$

and the solution of $A_{tr}c = b_{tr}$ is reduced to n floating point operations.

In order to compute b_{tr} (14), it is enough to select certain elements of a vector, whereby:

$$b_{tr_i} = \text{tr}(A_i^T X) = \sum_{j=1}^{n-i} X_{i+j,j},$$

if $A_i = G^i$ ($i = 1 : p$); or

$$b_{tr_i} = \text{tr}(A_{p+i}^T X) = \sum_{j=1}^{n-i} X_{j,j+i},$$

if $A_{p+i} = (G^T)^i$ ($i = 1 : q$).

With the new operations introduced to compute $A(c)$, J , A_{tr} , b_{tr} and to solve $A_{tr}c = b_{tr}$, MIII and LP can be redesigned in the MIII (MIII for Toeplitz matrices) and the LPT (LP for Toeplitz matrices) algorithms, respectively. The time complexities of MIII and LP are

$$T(n) = \left\{ \frac{38n^3}{3} + O(n^2) + k \{26n^3 + O(n^2)\} \right\} t_f,$$

and

$$T(n) = \left\{ \frac{53n^3}{3} + O(n^2) + k \{2n^4 + O(n^3)\} \right\} t_f,$$

respectively; whereas the time complexities of LPT and LPT are

$$T(n) = k \left\{ \frac{50n^3}{3} + O(n^2) \right\} t_f,$$

and

$$T(n) = \left\{ n^4 + k \left\{ \frac{56n^3}{3} + O(n^2) \right\} \right\} t_f,$$

respectively, being t_f the execution time for a single floating point operation. Both MIII and LP time complexities decrease one order of magnitude when the structure of Toeplitz matrices is used; some experimental results in Fig. 1 and Fig. 2 show this behaviour. As it can be seen the profit is more clear when n increases.

4. Design and implementation of a Parallel Solution for the IASVPT

Assuming that most of the vectors and matrices of the IASVPT are partitioned by blocks and cyclically distributed among the processors of a logical mesh of $P = P_r \times P_c$ processors, some operations of MIII and LPT algorithms have been parallelized using routines of ScaLAPACK, such as:

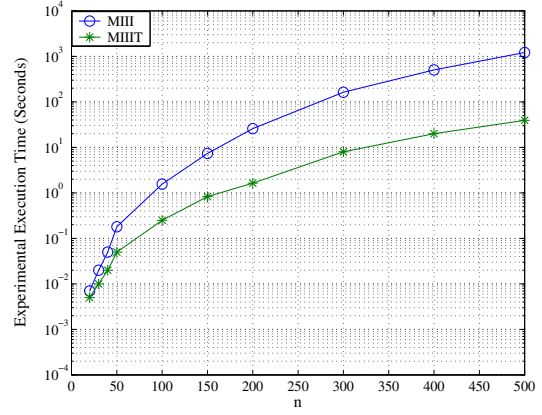


Fig. 1. Execution times of MIII vs MIII in logarithmic scale

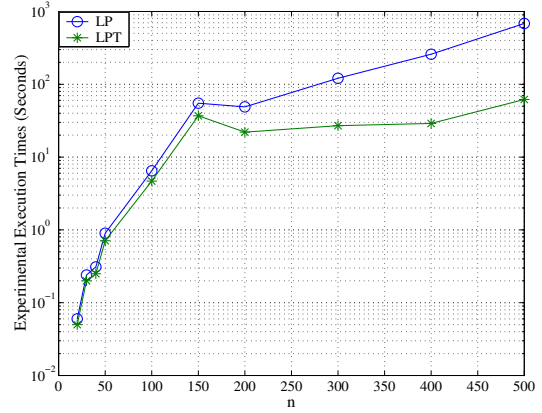


Fig. 2. Execution times of LP vs LPT in logarithmic scale

◇ The singular value decomposition (step 2 of Algorithm 1 and step 2.2 of Algorithm 2).

◇ The resolution of a linear system of equations (steps 3.2 and 3.5 of Algorithm 1).

Another parallel routines have been designed using routines of LAPACK, PBLAS and BLAS, implementing the communications with MPI and BLACS routines; these routines are:

◇ The parallel dot product to compute J (step 3.1. of Algorithm 1).

◇ The parallel copy of a vector to compute the product of a Toeplitz matrix \tilde{T}_i ($i = 1, 2, \dots, n$) by a vector, found in the computation of J and b_{tr} (step 2.3 of Algorithm 2).

◇ The parallel matrix–matrix product to compute b (step 3.1. of Algorithm 1), b_{tr} , \tilde{H} and \tilde{L} (step 3.4. of Algorithm 1), also to compute the stop criterion of MIII $\|U^T A(c)V - \Sigma^*\|_F$.

◇ The parallel vector–vector sum to compute b and the broadcast and reduce communication routines to redistribute and replicate c , given that c , S^* , S and \tilde{T}_0 are replicated in each processor to reduce the communications in some other operations, for example the computation of A (steps 1, 3.3 of Algorithm 1 and step 2.1 of Algorithm 2) and the LPT stop criterion $\|c^{(k)} - c^{(k-1)}\|_2$.

◇ The Frobenius–norm and 2–norm to compute the stop criterion of MIIIT and LPT.

The rest of MIIIT and LPT operations are scalar operations or they are limited to local data redistribution.

The time complexities of the resulting parallel algorithms are

$$T(n, P) = \left\{ O\left(\frac{n^3}{P}\right) + kO\left(\frac{n^3}{P}\right) \right\} t_f + \left\{ O\left(n \log \sqrt{P}\right) + kO\left(\sqrt{P} n^2\right) \right\} t_m + \left\{ O\left(\frac{n^2 \log P}{\sqrt{P}}\right) + kO\left(\frac{n^3 \log P}{\sqrt{P}}\right) \right\} t_v,$$

for parallel MIIIT (PMIIIT) and

$$T(n, P) = O\left(\frac{n^3}{P}\right) t_f + kO\left(n \log \sqrt{P}\right) t_m + kO\left(\frac{n^2 \log P}{\sqrt{P}}\right) t_v,$$

for parallel LPT (PLPT), where k is the number of iterations, t_m is the network latency and t_v is the inverse of the bandwidth. The time complexities of PMIIIT and PLPT are also smaller than that of the PMIII and the PLP algorithms:

$$T(n, P) = \left\{ O\left(\frac{n^3}{P}\right) + kO\left(\frac{n^4}{P}\right) \right\} t_f + \left\{ O\left(n \log \sqrt{P}\right) + kO\left(\sqrt{P} n^2\right) \right\} t_m + \left\{ O\left(\frac{n^2 \log P}{\sqrt{P}}\right) + kO\left(\frac{n^3 \log P}{\sqrt{P}}\right) \right\} t_v,$$

and

$$T(n, P) = \left\{ O\left(\frac{n^4}{P}\right) + kO\left(\frac{n^3}{P}\right) \right\} t_f + \left\{ O\left(n^2 \sqrt{P}\right) + kO\left(\sqrt{P} n\right) \right\} t_m + \left\{ O\left(\sqrt{P} n^2\right) + kO\left(\frac{n^2 \log P}{\sqrt{P}}\right) \right\} t_v,$$

respectively. Both PMIIIT and PLPT algorithms reduce the sequential execution time significantly using more than one processor (Fig. 3, Fig. 4).

In order to analyse the performance of PMIIIT and PLPT, the speedup ($S(n, P) = \frac{T(n, 1)}{T(n, P)}$) parameter [18] is experimentally measured. Fig. 5 shows how the

speedup of PMIIIT increases with the problem size n . The speedup for $n = 2000$ is quite good with 2 processors and acceptable with 4 processors. However, a better speedup is reached with PLPT (Fig. 6), in all the experiments the performance is good using different number of processors.

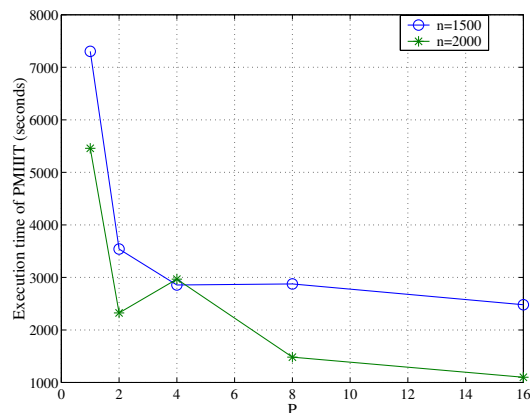


Fig. 3. Parallel execution time of PMIIIT versus number of processors

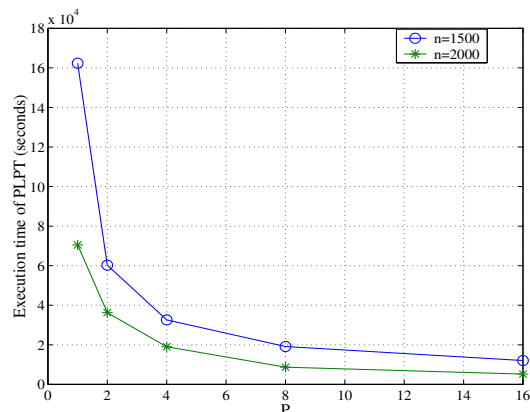


Fig. 4. Parallel execution time of PLPT versus number of processors

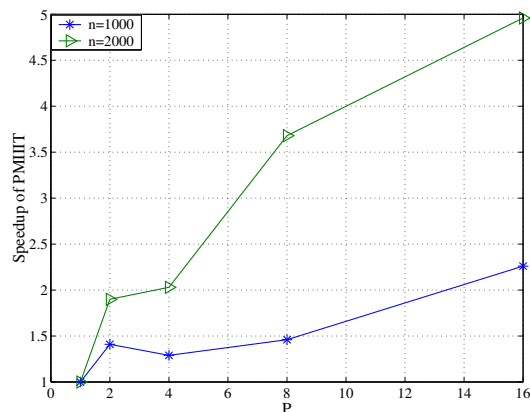


Fig. 5. Speedup of PMIIIT versus number of processors

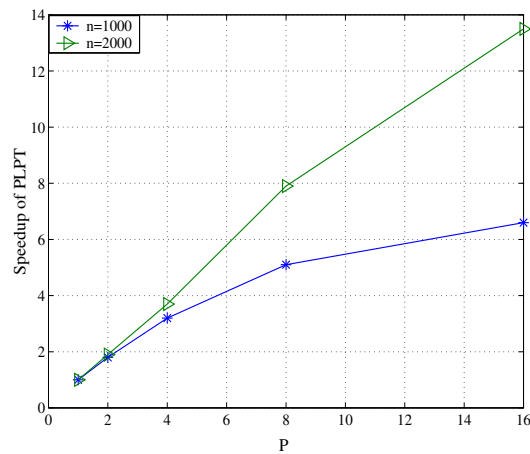


Fig. 6. Speedup of PLPT versus number of processors

5. Conclusions

Two efficient algorithms have been presented to solve the Inverse Additive Singular Value Problem in the case of Toeplitz matrices (IASVPT). These algorithms exploit the special structure of Toeplitz matrices, drastically reducing the storage requirements. The time complexities of the new algorithms are one order of magnitude less than the algorithms that solve the IASVP without using this feature. Experimental results show this improvement in time being more significant with large matrices.

The parallel algorithms implemented to solve the IASVPT make a considerable reduction of the sequential execution time. The performance achieved regarding the speedup is quite good under some conditions. PMIIIT improves its performance as the problem size increases. PLPT is a parallel algorithm that presents a good performance in all cases. In addition, our parallel algorithms take advantage of using efficient and portable linear algebra and communications libraries.

Acknowledgement

This work has been supported by Spanish MEC and FEDER under Grant TIC2003-08238-C02-02 and SEIT-DGEST-SUPERA-ANUIES (México).

References

- [1] Gramm, A.G.: *Inverse Problems: Mathematical and Analytical Techniques with Applications to Engineering*. Springer (2005)
- [2] Uhlmann, G., Levy, S. (Editors): *Inside Out: Inverse Problems and Applications*. Cambridge University Press (2003)
- [3] Lebedev, L.P., Vorovich, I.I., Gladwell, G.M.: *Functional Analysis: Applications in Mechanics and Inverse Problems*. Kluwer Academic Publishers (2002)
- [4] Chu, M.T.: Numerical Methods for Inverse Singular Value Problems. *SIAM, Journal Numerical Analysis*, Vol. 29, No. 3 (1992) 885-903

- [5] Flores-Becerra, G., García, V.M., Vidal, A.M.: Numerical Experiments on the Solution of the Inverse Additive Singular Value Problem. *Lecture Notes in Computer Science*, Vol. 3514, (2005) 17-24
- [6] Flores-Becerra, G., García, V.M., Vidal, A.M.: Parallelization of a Method for the Solution of the Inverse Additive Singular Value Problem. *WSEAS Transactions on Mathematics*, Vol. 5, No. 1 (2006) 81-88
- [7] Flores, G., Vidal, A.M.: Parallel Global and Local Convergent Algorithms for Solving the Inverse Additive Singular Value Problem. *WSEAS Transactions on Circuits and Systems*, Vol. 3, No. 10 (2004) 2241-2246
- [8] Hammarling, S., Dongarra, J., Du Croz, J., Hanson, R.J.: *An extended set of fortran basic linear algebra subroutines*. ACM Trans. Mathematical Software (1988)
- [9] Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J.: *LAPACK User Guide; Second edition*. SIAM (1995)
- [10] Choi, J., Dongarra, J., Ostrouchov, S., Petitt, A., Walker, D.: A proposal for a set of parallel basic linear algebra subprograms. Technical Report UT-CS-95-292, Department of Computer Science, University of Tennessee (1995)
- [11] Blackford, L.S., Choi, J., Clear, A.: *ScaLAPACK User's Guide*. SIAM (1997)
- [12] Dongarra, J., Van de Geijn, R.A.: Two dimensional basic linear algebra communications subprograms. Technical Report ST-CS-91-138, Department of Computer Science, University of Tennessee (1991)
- [13] Group, W., Lusk, E., Skjellum, A.: *Using MPI: Portable Parallel Programming with Message Passing Interface*. MIT Press (1994)
- [14] Chan, R., Bai, Z., Morini, B.: On the Convergence Rate of a Newton-Like Method for Inverse Eigenvalue and Inverse Singular Value Problems. *Int. J. Appl. Math.*, Vol. 13 (2003) 59-69
- [15] Chen, X., Chu, M.T.: On the Least Squares Solution of Inverse Eigenvalue Problems. *SIAM, Journal on Numerical Analysis*, Vol. 33, No. 6 (1996) 2417-2430
- [16] Brockett, R.W.: Dynamical systems that sort lists and solve the linear programming problems. *Linear Alg. Appl.*, Vol. 146 (1991) 79-91
- [17] Chu, M.T.: Matrix differential equations: A continuous realization process for linear algebra problems. *Nonlinear Anal., TMA*, Vol 1, No. 12 (1992) 1125-1146
- [18] Kumar, V., Grama, A., Gupta, A., Karypis, G.: *Introduction to Parallel Computing. Design and analysis algorithms*. The Benjamin/Cummings Publishing Company (1994)