Parallel Computation of the Eigenstructure of Toeplitz-plus-Hankel matrices on Multicomputers

José M. Badía * and Antonio M. Vidal *
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
46071 Valencia, Spain
E-mail: cpjmb@dsic.upv.es

Abstract

In this paper we present four parallel algorithms to compute any group of eigenvalues and eigenvectors of a Toeplitz-plus-Hankel matrix. These algorithms parallelize a method that combines the bisection technique with a fast root-finding procedure to obtain each eigenvalue. We design a parallel algorithm that applies a static distribution of the calculations among processors and three algorithms that use the farming technique to dynamically balance the load. All the algorithms have been implemented on a Multicomputer based on a network of transputers. We also present a study of the experimental performances and compare the different algorithms. The results obtained are in many cases very near to the maximum theoretical performances we can expect.

1. Introduction

The main goal of this paper is to describe four parallel algorithms so that we can efficiently obtain any group of eigenvalues and eigenvectors of Toeplitz-plus-Hankel matrices. Besides we present a detailed study of the experimental performances of these algorithms on a distributed memory multiprocessor.

This kind of matrices appears in signal and control theory when we apply boundary element methods [7], [8]. When we design parallel algorithms that deal with structured matrices we have to take into account two aspects of the problem. First, we try to parallelize sequential algorithms that exploit the structure of the matrices to obtain better performances. Second, we try to exploit the potential parallelism of the sequential method to obtain the maximum performance.

In this case we have parallelized an algorithm presented by Trench in [13]. This algorithm uses sequences of Sturm and combines bisection with a root-finding procedure to obtain each eigenvalue. This kind of technique has been widely applied to symmetric tridiagonal matrices [10] and more recently has been used with symmetric Toeplitz matrices [12] and with banded symmetric Toeplitz matrices [14].

As far as the parallelization of the sequential method is concerned, we have used some ideas that have been previously applied to tridiagonal matrices [2], [6] and to symmetric Toeplitz matrices [1] and we have adapted the method to the case of Toeplitz-plus-Hankel matrices. As we will see in the following sections, we have used basically two schemes: a technique with a static distribution of the computations and a farming technique to dynamically balance the load.

In section 2 we briefly analyze the theoretical basis of the sequential method and we outline the main ideas of the sequential algorithm. In section 3 we describe the parallel machine we have used and we show the experimental results of the sequential algorithm. In section 4 we describe the parallel algorithms and we present the results when we run

* Partially supported by the ESPRIT III Basic Research Programm of the EC under contract No. 9072 (Project GEPPCOM), and partially supported by Spanish CICYT project TIC-91-1157-C03-02

them on a multicomputer. Finally, in section 5 we analyze the experimental performances and we compare the parallel algorithms that we have implemented.

2. Theoretical basis and sequential algorithm

Given a symmetric matrix $A_n \in R^{nxn}$ that is the result of adding a symmetric Toeplitz matrix T_n and a Hankel matrix H_n, our objective is to compute any group of eigenvalues and eigenvectors of A_n.

First we recall what we mean by Toeplitz and by Hankel matrix. From the point of view of its structure, a Toeplitz matrix is such that all its entries are constant along each diagonal. On the other hand, a Hankel matrix has its entries constant along each northeast-southwest diagonal. Therefore it is possible to determine both kinds of matrices from a single vector. In the case of a symmetric Toeplitz matrix we use the vector

$t = [t_0, t_1, \ldots, t_{n-1}]^T$

and in the case of a Hankel matrix we use the vector

$h = [h_0, h_1, \ldots, h_{2n-1}]^T$

We can then define the matrix $A_n = T_n + H_n$ as the matrix whose elements are

$$a_{i,j} = t_{|i-j|} + h_{(i+j-2)} \qquad i,j : 1, 2, \ldots, n$$

The basic idea of the sequential algorithm is to use sequences of Sturm to delimit the position of every eigenvalue on the real axis. Given that A_n is a symmetric matrix all its eigenvalues are real.

To define the sequences we will refer to

$$A_m = \left(a_{i,j}\right)_{i,j=1}^{m} \qquad 1 \le m \le n$$

where every A_m is the leading submatrix of A_n of size mxm. Using these matrices we define the sequence of Sturm of A_n in λ as

$p_0(\lambda) = 1$

$p_m(\lambda) = \det(A_m - \lambda I_m) \qquad 1 \le m \le n$

Related to the sequence of Sturm we can define the following sequence

$$q_m(\lambda) = \frac{p_m(\lambda)}{p_{m-1}(\lambda)} \qquad 1 \le m \le n \qquad (1)$$

As it can bee seen in [13] there is a lower unit triangular matrix $L_m(\lambda)$ such that

$$L_m(\lambda)\left(A_m - \lambda I_m\right)L_m^T(\lambda) = \mathrm{diag}\left(q_1(\lambda), q_2(\lambda), \cdots, q_m(\lambda)\right)$$

Thus, from the Sylvester Law of Inertia [3] it occurs that the number of negative elements of the sequence (1) is equal to the number of eigenvalues of A_n smaller than λ.

Then it is possible to apply a bisection technique that uses the sequence (1) to obtain any eigenvalue of a Toeplitz-plus-Hankel matrix. In fact it is possible to apply this method to compute the eigenvalues of any symmetric matrix, but the cost to obtain each eigenvalue in the general case is $O(n^3)$.

However Trench [13] includes the Toeplitz-plus-Hankel matrices in a more general type of matrices called ESH (Efficiently Structured Hermitian). These matrices enable us to obtain each eigenvalue with a cost $O(n^2)$ using the sequences in (1). A matrix is ESH if it is possible to compute the last column of all the inverse matrices $(A_m - \lambda I_m)^{-1}$ $(1 \le m \le n)$ with a cost $O(n^2)$. The Toeplitz-plus-Hankel matrices comply with this condition. We can also include as ESH matrices the symmetric Toeplitz ones.

The property mentioned above enables us to use a "fast" algorithm to compute the inverse of a Toeplitz-plus-Hankel matrix given by Heining, Jankowski and Rost [4]. This algorithm allows computing the last column of $(A_n - \lambda I_n)^{-1}$ using five recurrence formulae and with a cost $O(n^2)$. Moreover this algorithm allows us to obtain the elements of the sequence (1) during the inversion process.

Let us outline the main ideas of the sequential algorithm. We will use two results related to the sequence (1). First the number of negative elements in this sequence will server us to isolate the eigenvalues. Second the we will use the value of $q_n(\lambda)$ whose zeros are the eigenvalues of A_n. The last result will allow us to extract each eigenvalue with any accuracy allowed by the machine.

To obtain a single eigenvalue λ_k we use an interval (a, b) that contains it and that can be obtained using the Gershgoring circles of A_n [3]. We first perform the isolation phase using repeated bisection to compute an interval (a_k, a_{k+1}) that only contains the eigenvalue λ_k of A_n and no other eigenvalue of A_n or A_{n-1}. With this condition $q_n(\lambda)$ is continuos in this interval. In a second phase of extraction we use a root-finding procedure which is faster than the plain bisection to compute the eigenvalue as the zeros of the function $q_n(\lambda)$ in the interval (a_k, a_{k+1}).

This method also allows us to compute any group of eigenvalues. In this case we combine the technique we have previously described with a queue of intervals. We begin by defining an interval (a, b) that contains all the eigenvalues to compute using Gershgoring circles. Next, we apply repeated bisection to isolate the first eigenvalue we are looking for. During this process of bisection we store in the queue the intervals generated that contain at least one eigenvalue. Once we have isolated the first eigenvalue, we proceed to its extraction using a root-finding procedure just as we did when we wanted to obtain a single eigenvalue. Once this eigenvalue has been obtained we take a new interval of the queue and we repeat the whole process to compute its first eigenvalue. This technique guarantees that we can obtain any group of eigenvalues in a finite number of steps and the process finishes when the queue is empty. An algorithmic description of this method in the case of symmetric Toeplitz matrices can be found in [1].

Regarding the computation of the associated eigenvectors, the algorithm that computes the sequence (1) provides us, without any additional cost, the eigenvector from one of the five recurrence formulae mentioned above.

3. Parallel architecture and sequential results

We have used a distributed memory multiprocessor to implement both the sequential and all the parallel algorithms. The Parsys SN-1000 or Supernode [9] is a multicomputer based on a network of T800 transputers. The topology of the interconnection network can be defined by the user by software. The implementation of all the algorithms on this machine has been performed using the parallel language Occam-2 [5] that was specifically created to program the transputer microprocessor.

To obtain the results of the sequential algorithm we have used a single T800 transputer. In all the executions we obtain all the eigenvalues of matrices randomly generated. In this paper we will only deal with aspects regarding the cost of the algorithm. A detailed analysis of the numerical stability of the sequential method can be found in [13].

The results of all the tables are in seconds and represent the mean time to obtain all the eigenvalues of 3 matrices of each size. The analysis has been carried out with matrices of sizes 32, 64, 128 and 256.

32	**64**	**128**	**256**
22.5	186.2	1417.2	11416.2

Figure 1. Times in seconds for the sequential algorithm.

4. Parallel algorithms

4.1. Parallelism of the method

In this section we briefly describe the different types of parallelism that we can apply to the sequential method of section 2.

First, we could try to parallelize the algorithm to compute the sequence (1) for a given value of λ. However this algorithm basically deals with recurrence formulae and thus it is difficult to parallelize. Second, all the processors could cooperate to obtain the sequence (1) for different values of λ. We can then use a multisection method instead of the bisection method. Using the third type of parallelism every processor could compute a different group of eigenvalues in parallel with the rest.

When we are trying to obtain a large group of eigenvalues the third type of parallelism is the most appropriate and thus this is the parallelism that we use in all our algorithms.

4.2. Algorithm with static load balance. SEP

The basic idea of this algorithm is to exploit the fact that the computation of each eigenvalue can be performed independently of the computation of the rest. Basically, we distribute the computation of a group of the eigenvalues on each processor.

We have used a bidirectional array topology to implement the algorithm. First we define the initial interval that contains all the eigenvalues to compute using Gershgoring circles. Next we distribute the computation of the eigenvalues among the processor in such a way that each processor computes the same number of eigenvalues or at most one more than the rest. Then, every processor applies to the initial interval the sequential method of section 2 to compute its eigenvalues.

The following table shows the time in seconds that we need to compute all the eigenvalues of the same matrices randomly generated used in the case of the sequential algorithm. The results have been obtained using bidirectional arrays with 2, 4, 8 and 16 processors.

	2	**4**	**8**	**16**
32	11,97	6,58	3,82	2,63
64	96,04	50,57	29,16	18,03
128	748,56	385,22	206,92	123,84
256	5765,34	2940,22	1558,37	905,35

Figure 2. Times in seconds of the SEP algorithm.

4.3. Algorithm with dynamic load balance. FARM

The algorithm with static distribution we have presented in the previous section distributes the computation of the same number of eigenvalues among each processor. However this does not guarantee that all the processors perform the same quantity of operations. In fact this algorithm does not obtain the maximum performances due to a certain load unbalance.

A possible solution of this problem is to dynamically distribute the load among the processors. We have used a farming technique to achieve this distribution. This method has been already applied to the case of tridiagonal matrices [11] and more recently to the case of symmetric Toeplitz matrices [1].

We have also used a bidirectional array topology to implement this algorithm. The first processor, that we will call farmer, is in charge of distributing the calculations among the rest of processors of the farm, the workers. The farmer manages a queue of intervals containing at least one eigenvalue and it sends intervals to the farm each time a processor has finished the computation of an eigenvalue. As far as the rest of processors are concerned they wait an interval, then they apply the sequential procedure to isolate and extract its first eigenvalue and finally they send the result to the farmer indicating that they are ready to receive a new interval. During the bisection process the non empty intervals produced are sent to the farmer and this processor stores the intervals in the queue to redistribute them dynamically.

The next table shows the results obtained with this algorithm in the same cases used for the SEP algorithm

	2	4	8	16
32	12,48	6,68	3,85	2,71
64	100,02	51,26	27,35	16,21
128	755,98	384,07	197,97	110,45
256	6070,16	3060,60	1576,05	830,66

Figure 3. Times in seconds of the FARM algorithm.

The following figure allows us to study the load balance achieved with the SEP and FARM algorithms. In this figure we can see the number of "iterations", that is, the number of executions of the algorithm to compute the sequence (1), that every processor performs. We present the result in the case of a matrix of size 256 and using 16 processors. We can see that the SEP algorithm has a certain load unbalance in the number of iterations although every processor computes the same number of eigenvalues. However this problem is notably reduced in the FARM algorithm showing the advantage of using a dynamic distribution technique.

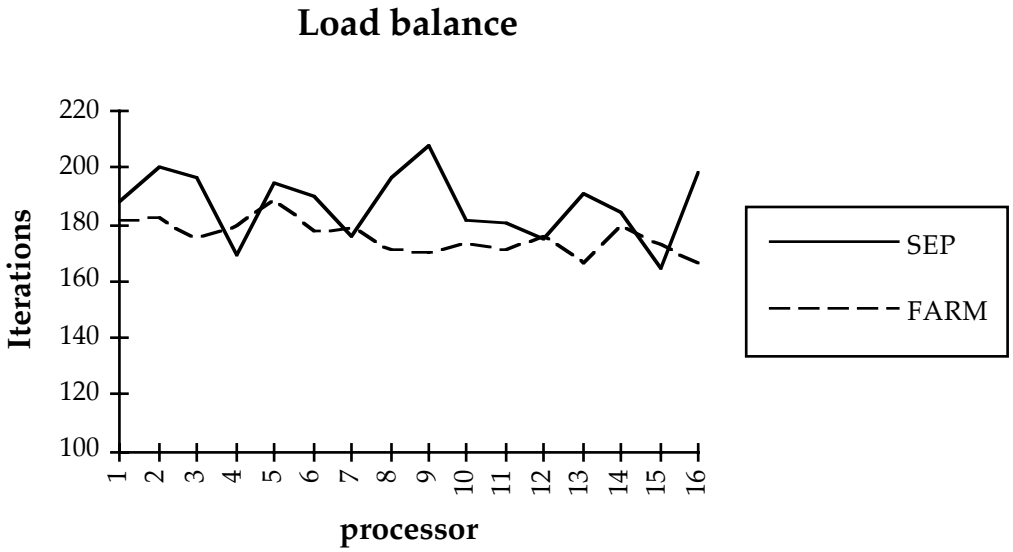

Figure 4. Load balance among the processors. p=16, n=256.

4.4. Algorithms with multiple farms

Although the FARM algorithm efficiently balances the load when we work with large matrices, it presents some problems when we use long arrays of processors and small matrices. These problems occur because the processors closer to the farmer compute the main part of the eigenvalues and the last processors, in the worst case, remain idle during all the process.

The solution to this problem is to reduce the length of the arrays using several farms connected to different links of the farmer. Using this strategy the processor farmer concurrently manages the different farms and distributes the calculations among their processors. Every processor worker executes the same process that in the case of the FARM algorithm. We have implemented two algorithms using this technique, respectively the DFARM algorithm that uses two farms and the TFARM algorithm that uses three farms.

The following two tables show the results achieved with these algorithms when we compute all the eigenvalues of the same matrices used for the previous algorithms.

	2	4	8	16
32	12.17	6.53	3.84	2.57
64	99.33	51.02	27.15	15.86
128	754.90	382.19	197.81	110.01
256	6068.99	3064.10	1576.82	830.55

Figure 5. Times in seconds for the DFARM algorithm.

	4	8	16
32	6,41	3,72	2,59
64	50,81	27,15	15,68
128	381,94	197,69	109,48
256	3065,78	1574,17	828,85

Figure 6. Times in seconds for the TFARM algorithm.

5. Comparison of performances

We use the speedup in order to analyze and compare the experimental performances of the algorithms. This parameter enables us to study the improvement achieved using the parallel algorithm over the sequential version executed in one transputer.

	2		4		8		16	
	SEP	**FARM**	**SEP**	**FARM**	**SEP**	**FARM**	**SEP**	**FARM**
32	1.88	1.81	3.43	3.37	5.90	5.85	8.59	8.32
64	1.94	1.86	3.68	3.63	6.39	6.81	10.33	11.49
128	1.89	1.87	3.68	3.69	6.85	7.16	11.44	12.83
256	1.98	1.88	3.88	3.73	7.33	7.24	12.61	13.74

Figure 7. Speedups of the SEP and FARM algorithms.

	2	4		8		16	
	DFARM	**DFARM**	**TFARM**	**DFARM**	**TFARM**	**DFARM**	**TFARM**
32	1.85	3.45	3.52	5.87	6.06	8.76	8.70
64	1.87	3.65	3.66	6.86	6.86	11.74	11.87
128	1.88	3.71	3.71	7.16	7.17	12.88	12.94
256	1.88	3.73	3.72	7.24	7.25	13.75	13.77

Figure 8. Speedups of the DFARM and TFARM algorithms.

The two previous tables show the speedups of the parallel algorithms obtained from the execution times we have shown in sections 3 and 4. The first aspect we can point out is that the results are quite close to the theoretical maximum in the majority of the cases. This effect is more notable with large matrices and with short arrays of processors.

Regarding the comparison of the parallel algorithm, the next figure serves us to compare its behaviour when we modify the size of the matrices. In this case we represent the speedups obtained using 16 processors.

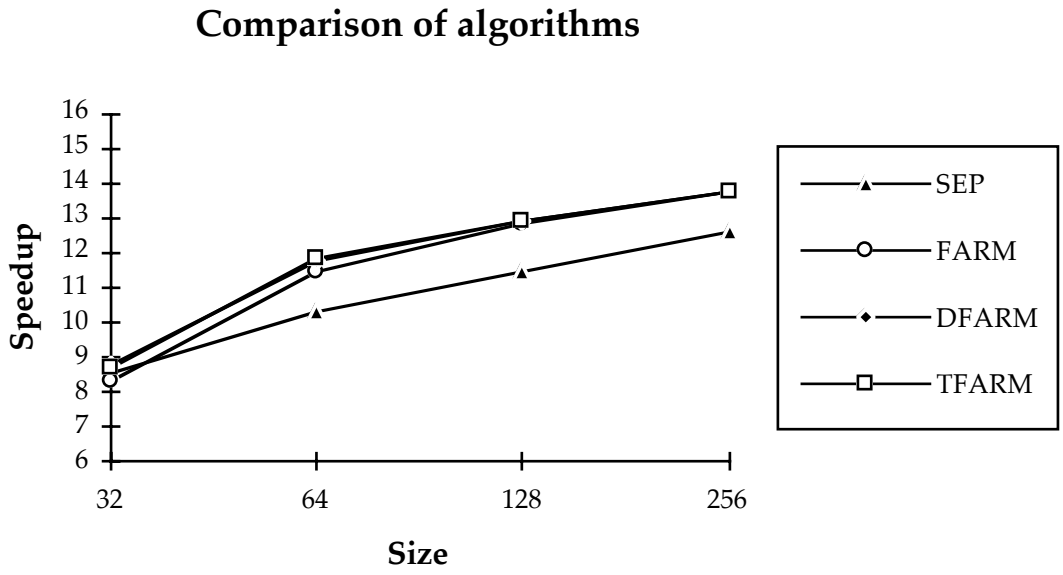

Figure 9. Comparison of the algorithms modifying the size.

We can see that the results achieved with the SEP algorithm are worse than those obtained with the three algorithms that use the farming technique. This is due to the better load balance achieved in the last case as we showed in figure 4. As far as the three algorithms of type farm are concerned, they achieve very similar performances although in the majority of the cases we obtain better results using several farms.

6. Conclusions

In this paper we have verified that it is possible to efficiently parallelize a method to obtain any group of eigenvalues and eigenvectors of Toeplitz-plus-Hankel matrices.

The use of a distributed memory multiprocessor and the appropriate distribution of the computations among the processors allows us to obtain experimental performances close to the theoretical maximum ones.

The analysis of the experimental results shows that the dynamic distribution of the computations improves the performances of a static distribution, even though it introduces communications in the algorithm.

Acknowledgement

The authors thank professor William Trench for providing them the software of the sequential algorithm that has served as the basis of the parallel algorithms implemented in this paper.

7. References

[1] Badía, J. M. & Vidal, A. M., "Divide and Conquer Algorithms to solve the Eigenproblem of Symmetric Toeplitz Matrices on Multicomputers", DSIC-II/2/94 ,Dpt. Sistemas Informáticos y Computación. U.P.V. (1994).

[2] Bernstein, H. J. & Goldstein, M., "Parallel Implementation of Bisection for the Calculation of Eigenvalues of Tridiagonal Symmetric Matrices.", Computing, no. 37. pp. 85-91 (1986).

[3] Golub, G. H. & Van Loan, C. F., "Matrix Computations.", Johns Hopkins University Press, Eds., (1989).

[4] Heinig, G.; Jankowski, P. & Rost, K., "Fast Inversion Algorithms of Toeplitz-plus-Hankel Matrices.", Numer. Math., no. 52. pp. 665-682 (1988).

[5] INMOS, "Occam 2. Reference Manual.", Prentice Hall International Eds., Series in Computer Science 1989.

[6] Kalamboukis, T. Z., "The Symmetric Tridiagonal Eigenvalue Problem on a Transputer Network.", Parallel Computing, no. 15. pp. 101-106 (1990).
[7] Ljung, S., "Fast algorithms for integral equations and least square identification problem", Diss. 93 ,Linköp Stud. Sc. Tech. (1983).
[8] Merchant, G. A. & Parks, T. W., "Efficient solution of a Toeplitz-plus-Hankel coefficient matrix system of equation.", IEEE Trans. on Acoust. Speech Sign. Process, vol. 30. no. 1. pp. 40-44 (1982).
[9] PARSYS, "Hardware Reference for the Parsys SN1000 series 1989.
[10] Pereyra, V. & Scherer, G., "Eigenvalues of Symmetric Tridiagonal Matrices: A Fast, Accurate and Reliable Algorithm.", J. Inst. Math. Applics, no. 12. pp. 209-222 (1973).
[11] Ralha, R. M. S., "Parallel Solution of the Symmetric Tridiagonal Eigenvalue Problem on a Transputer Network", SEMNI 93, La Coruña, Spain, 1993, vol. 2, pp. 1026-1033.
[12] Trench, W. F., "Numerical Solution of the Eigenvalue Problem for Hermitian Toeplitz Matrices.", SIAM J. Matrix Anal. Appl., vol. 10. no. 2. pp. 135-146 (1989).
[13] Trench, W. F., "Numerical Solution of the Eigenvalue Problem for Efficiently Structured Hermitian Matrices.", Linear Algebra and its Applications, no. 154-156. pp. 415-432 (1991).
[14] Trench, W. F., "A Note on Computing Eigenvalues of Banded Hermitian Toeplitz Matrices.", SIAM J. Sci. Stat. Comput, vol. 14. no. 2. pp. (1993).