15th Triennial World Congress, Barcelona, Spain

PARALLELSLICOT MODEL REDUCTION ROUTINES: THE CHOLESKY FACTOR OF GRAMMIANS

David Guerrero * **Vicente Hernández*** **José E. Román***

* *DSIC - UPV, Camino de Vera s/n, Valencia, Spain*

Abstract: This paper presents part of the work being carried out to obtain parallel versions of the main SLICOT routines for model reduction. It is focused on the parallel solution of standard Lyapunov equations obtaining the Cholesky factor of the controllability and observability Grammians. This operation is an important basis for model reduction methods. Routines from the standard libraries BLAS, LAPACK, SLICOT, PBLAS and ScaLAPACK have been used whenever possible in the parallelisation process. However, it has been necessary to develop some new routines. Experimental results obtained using a cluster of PC's are shown.

Keywords: Parallel algorithms, Model reduction, Linear control systems, Lyapunov equation, Cholesky factorisation

1. INTRODUCTION

High order systems appear commonly in control problems. They allow to increase precision of the results in control or simulation processes. However, a larger size of the system involves higher resources needs, both in memory and computing terms. Thus, there is a necessity of reducing high order models of linear control systems.

Reducing a model consists in obtaining a lower order model with a similar behaviour. By using this reduced model in the control or simulation process, necessary requirements are decreased. But the process of obtaining the reduced order model is also an expensive process itself, since it has to work with the original unreduced system. Thus, it is convenient to develop parallel algorithms of the main reduction methods to decrease the time involved in the process.

SLICOT is a control library which incorporates several model reduction routines. It is freely available and it has many desirable features for a standard control library . Thus, it is a suitable sequential starting point. It implements most of the model reduction methods used, such as those based on *Square-Root, Balancing-Free Square-Root, Singular Perturbation Approximation* and *Hankel-Norm Approximation.*

One important operation in these model reduction methods is the solution of standard Lyapunov equations. Developing a parallel solver for this equation is a first step to parallelise the full process of model reduction.

There are several sequential implementations to solve this equation (Penzl, 1996). However, the parallel case has not been explored in such an extended way. This paper shows the work carried out to obtain a parallel solver for the standard Lyapunov equation following the Hammarling's method (Hammarling, 1982).

In the next sections, first a brief description of the main model reduction methods is given. Then, the SLICOT library is presented as part of the NICONET project. Later, model reduction routines of the SLICOT library are enumerated. The method used to solve standard Lyapunov equations is explained, and then the parallel approximation is described. Finally, some experimental results obtained in a cluster of PC's are shown.

2. MODEL REDUCTION METHODS BASED ON BALANCING TECHNIQUES

Model reduction methods based on balancing techniques are appropiate to be applied to stable systems. These methods rely on the same basic operations, thus allowing to develop them by implementing these basic operations. Some of these methods based on or related to balancing techniques are implemented in SLICOT (Varga, 1999).

Moreover, the basic methods combined with coprime factorisation or spectral decomposition techniques can be used to reduce unstable systems or to perform frequency-weighted model reduction (Varga, 1999). Implementing the methods for stable systems can be seen as a previous step in the process of obtaining general model reduction codes for general systems.

Starting from an n-th order original state-space model $G = (A, B, C, D)$ with corresponding transfer-function matrix (TFM) $G(\lambda) = C(\lambda I - A)^{-1}B + D$, the goal is to obtain an r-th order approximation $G_r = (A_r, B_r, C_r, D_r)$ of the original model (with $r < n$) with TFM $G_r = C_r(\lambda I - A_r)^{-1}B_r + D_r$. Many model reduction methods can be seen as computing a similarity transformation Z which leads to

$$\left[\begin{array}{c|c} Z^{-1}AZ & Z^{-1}B \\ \hline CZ & D \end{array}\right] = \left[\begin{array}{cc|c} A_{11} & A_{12} & B_1 \\ A_{21} & A_{22} & B_2 \\ \hline C_1 & C_2 & D \end{array}\right]. \quad (1)$$

Then, a reduced model (A_r, B_r, C_r, D_r) can be formed by taking $A_r = A_{11}$, $B_r = C_1$, $C_r = C_1$ and $D_r = D$.

If matrix Z used in the transformation is chosen as $Z = [T, U]$, $Z^{-1} = [L^T, V^T]^T$ with $LT = I_r$, the reduced system is (LAT, LB, CT, D).

A *singular perturbation approximation* (SPA) can be formed using the expressions

$$A_r = A_{11} + A_{12}(\gamma I - A_{22})^{-1}A_{21}, \quad (2)$$
$$B_r = B_1 + A_{12}(\gamma I - A_{22})^{-1}B_2, \quad (3)$$
$$C_r = C_1 + C_2(\gamma I - A_{22})^{-1}A_{21}, \quad (4)$$
$$D_r = D + C_2(\gamma I - A_{22})^{-1}B_2, \quad (5)$$

being $\gamma = 0$ for the continuous-time case and $\gamma = 1$ for the discrete-time case.

Frequently, matrices L and T are computed from two positive semi-definite matrices P and Q which have special representation of the properties of the system. These matrices correspond to the *Grammians* of the system. They can be obtained in the form of Cholesky factorisation, that is $P = S^T S$ and $Q = R^T R$. Then, L and T matrices can be computed by obtaining the *singular value decomposition* (SVD)

$$SR^T = [U_1 U_2]\mathrm{diag}(\Sigma_1, \Sigma_2)[V_1 V_2]^T, \quad (6)$$

with $\Sigma_1 = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$, $\Sigma_2 = \mathrm{diag}(\sigma_{r+1}, \ldots, \sigma_n)$ and $\sigma_1 \geq \ldots \geq \sigma_r > \sigma_{r+1} \geq \ldots \geq \sigma_n \geq 0$.

The $square-root$ (SR) method determines L and T as

$$L = \Sigma_1^{-\frac{1}{2}} V_1^T R, \;\; T = S^T U_1 \Sigma_1^{-\frac{1}{2}}. \quad (7)$$

The use of these matrices for the transformation leads to a system in which the Grammians are diagonal and equal. This is called a *balanced* realisation of the system. The SR method is appropiate for well-equilibrated systems. However, if the original system is highly unbalanced, the reduced model can suffer a lost of accuracy. To avoid this, a new approach called *balancing-free* (BF) has been proposed. However, this new approach can be less accurate for moderately ill-balanced systems. A *balancing-free square-root* (BFSR) approach has been introduced, which combines the advantages of the BF and SR approaches (Varga, 1999).

3. SEQUENTIAL ROUTINES FOR MODEL REDUCTION

3.1 *NICONET*

NICONET (www.win.tue.nl/niconet, Numerics in Control Network) is an European thematic network project with the aim of formalising and extending current collaboration with respect to robust numerical software for control systems analysis and synthesis. Activities in the past have resulted in two control libraries, SLICOT and RASP, that are still in active development.

The research within the NICONET project is organised in five tasks:

- I: Basic Numerical Tools for Control.
- II: Model Reduction.
  - II.A: Development of standard software for model reduction.
  - II.B: Development of standard software for controller reduction.
  - II.C: Development of standard software for model reduction of high order systems.
- III: Subspace Identification.
- IV: Robust Control.
- V: Nonlinear Systems in Robotics.

This work is part of the Task II.C, which is in charge of developing standard software for model reduction of high order systems. Dealing with high order systems involves the necessity of high performance computing techniques to be applied, such as parallel computing, to reduce time and memory requirements.

Both direct methods and iterative methods are used in Task II.C. This paper is oriented to the direct methods approach since it uses as a main operation the Schur decomposition. Iterative methods are based on the matrix sign and disk functions. Work being performed with iterative methods is described in (Benner *et al.*, 1999).

3.2 *SLICOT*

The freeware subroutine library SLICOT provides Fortran 77 implementations of numerical algorithms for computations in systems and control theory. Based on numerical linear algebra routines from BLAS and LAPACK (Anderson *et al.*, 1992), SLICOT provides methods for the design and analysis of control systems. The basic ideas behind the library are usefulness of algorithms, robustness, numerical stability and accuracy, performance with respect to speed and memory requirements, portability and reusability, standardisation and benchmarking.

The current version of SLICOT consists of about 250 user callable routines in various domains of systems and control. These routines have associated on-line documentation. About 150 routines have associated example programs, data and results. New routines are still in preparation. Due to the use of Fortran 77, reusability of the software is obtained, so SLICOT can serve as the core for various existing and future CACSD platforms and production quality software. SLICOT routines can be linked to MATLAB through a gateway compiler, e.g. the NAG Gateway Generator. SLICOT is freely available through anonymous ftp (wgs.esat.kuleuven.ac.be/pub/WGS/SLICOT/).

3.3 *Routines for model reduction of stable systems*

These are the main routines of the SLICOT library for model reduction of stable systems:

- `AB09AD` computes reduced (or minimal) order balanced models using either the Square-Root or the Balancing-Free Square-Root Balance & Truncate method.
- `AB09BD` computes reduced order models using the Balancing-Free Square-Root Singular Perturbation Approximation method.
- `AB09CD` computes reduced order models using the optimal Hankel-Norm Approximation method based on Square-Root balancing.
- `AB09DD` computes a reduced order model by using the Singular Perturbation formulas.

The work described in this paper has been oriented to parallelising routines `AB09AD`, `AB09BD` and `AB09DD`, whose main operations are the solution of Lyapunov equations (obtaining the Cholesky factor of the solution) and the singular value decomposition.

4. HAMMARLING'S METHOD FOR LYAPUNOV EQUATIONS

4.1 *Standard Lyapunov equation*

The standard Lyapunov equation is presented in two forms, one for continuous-time systems (8) and one for discrete-time systems (9):

$$A^T X + XA = -E, \tag{8}$$

$$A^T XA - X = -E. \tag{9}$$

Here A and E are real square matrices of size n. Matrix E is symmetric, as well as the solution matrix X when unique.

The most typical method used to solve these equations is due to Bartels and Stewart (Bartels and Stewart, 1972). It is based on transforming the equation to a reduced form, solving this reduced form and then transforming back the solution.

4.2 *Hammarling's method*

Hammarling's method (Hammarling, 1982) is an alternative for solving Lyapunov equations when their right-hand side is positive semidefinite and matrix A is stable. In these cases, the right-hand side of the equation is usually in the form of the product of a matrix by its transposed matrix, and the Cholesky factor U of the solution matrix X is the desired output. The above equations become

$$A^T U^T U + U^T UA = -B^T B, \tag{10}$$

$$A^T U^T UA - U^T U = -C^T C. \tag{11}$$

This method allows to obtain the Cholesky factor of the solution directly without computing explicitly the product in the right-hand side of the equation. It is similar to Bartels-Stewart method in that both of them work by transforming the equation to a reduced form, then solving this reduced equation and later obtaining the solution to the original equation by a back transformation. These steps are going to be described in detail for the continuous case of Hammarling's method.

Transformation to reduced form

The transformation to reduced form is performed to obtain the equation

$$A_s^T X_s + X_s A_s = -B_s^T B_s, \tag{12}$$

where B_s is an upper triangular matrix of size $n \times n$, and A_s is an upper quasi-triangular matrix

of the same dimensions. This form of the equation is obtained by reducing matrix A to the real Schur form A_s, via computing the orthogonal matrix Z which verifies $A_s = Z^T A Z$, being A_s an upper quasi-triangular matrix, that is, upper triangular with the exception of some nonzero elements in the first subdiagonal. A_s can be seen as an upper block triangular matrix, whose diagonal is formed by 1×1 or 2×2 blocks corresponding to real or complex eigenvalues of A, respectively.

In this transformation to reduced form (12), matrix B_s is obtained as the upper triangular matrix R from the QR factorisation of the product BZ.

Solution of the reduced equation

To solve equation (12) for U_s, Cholesky factor of the solution ($X_s = U_s^T U_s$), the involved matrices A_s, B_s and U_s are partitioned in a way that yields a 2 by 2 or 1 by 1 Lyapunov equation and two other equations. This small Lyapunov equation is solved as a linear system of equations (by Kronecker products). Then the other two equations are updated. One of them is solved as a reduced Sylvester equation in which involved matrices are upper quasi-triangular. The other equation is transformed by a QR decomposition and matrix products to a reduced Lyapunov equation which can be solved by this same procedure.

The solution of this reduced equation in parallel has been treated previously for the generalised case (Guerrero *et al.*, 1998) and for the standard case (Guerrero *et al.*, 2001).

Back transformation

Once the solution of the reduced equation has been computed, another transformation is required to obtain the solution of the original equation. In Hammarling's method, this transformation consists in obtaining the QR factorisation of the product of matrix U_s, solution of the reduced equation, and the transpose of matrix Z, coming from the transformation to real Schur form, that is, computing $U_s Z^T = Q_U U$.

With this transformation, the upper triangular matrix U, Cholesky factor of the solution X of equation (8), is obtained.

5. PARALLEL IMPLEMENTATION

The procedure to obtain a parallel code for the complete solution of Lyapunov equations by Hammarling's method is based on using available parallel routines for each of the basic operations performed in every step. However, some of those operations had not still been implemented in parallel. New parallel routines have been developed for them.

Next, the new parallel routines developed by the authors of this paper are described briefly. The names of these routines are in bold type in the text.

PDLPHM is the main routine to be called for solving Lyapunov equations by Hammarling's method. It allows to solve both continuous and discrete time Lyapunov equations, as well as their transpose versions.

The routine performs the reduction of the equation, solving of it, and back transformation of the final solution, using new developed routines.

PDGEES routine obtains the real Schur form of a generic matrix. The resulting Schur form is in standard form, that is, the 2×2 blocks which may appear in the main block diagonal are in the form $\begin{bmatrix} a & b \\ c & a \end{bmatrix}$ where $b * c < 0$, being $a \pm \sqrt{bc}$ the corresponding eigenvalues of that block.

PDORGHR routine converts the input orthogonal matrix stored as reflectors and τ values to a regular matrix.

PDLANV routine has been required since the routine PDLAHQR currently available in ScaLAPACK does not return the 2×2 blocks of the real Schur form matrix in standard form. This routine converts these blocks into standard form and it also accumulates in the appropriate way the used transformations in the orthogonal matrix which represents the operations performed to transform the original matrix to Schur form.

PDGEMM2 routine performs the operation $B \leftarrow \mathrm{op}(A) \times B$ or $B \leftarrow B \times \mathrm{op}(A)$, where $\mathrm{op}(A)$ is A or A^T, i.e. it computes the product of two general matrices overwriting one of them with the result.

PDSHIFT routine performs a horizontal displacement of a matrix. It is needed in the transpose case of the equation if the original matrix B of the right hand side is not square.

PDDIAGZ routine is used several times in the code to make the elements in the main diagonal of a matrix non-negative by scaling with minus one the rows (or columns) corresponding to negative diagonal elements. Trapezoidal matrices are allowed.

PSB03OT routine is a parallel version of the SLICOT routine SB03OT, which is used to solve a *reduced* Lyapunov equation obtaining the Cholesky factor of the solution. More information about this routine can be found in (Guerrero *et al.*, 2001).

PDTRSCAL routine allows to scale a triangular (or trapezoidal) distributed matrix by a value.

PSB03OR is a parallel version of the SLICOT routine SB03OR. In the same way that SB03OR is for SB03OT, **PSB03OR** is a *service routine* for **PSB03OT**, that is a routine developed to complement the need of solving a type of Sylvester equation in **PSB03OT**. **PSB03OR** allows to solve in parallel Sylvester equations whose matrices are in Schur form.

PMB04OD *and* **PMB04ND** routines are parallel versions of the SLICOT routines MB04OD and MB04ND. However, they are light versions in the sense that the parallel versions do not have all the functionality offered by sequential ones. They have only the functionality required by **PSB03OT**. These routines perform QR (**PMB04OD**) or RQ (**PMB04ND**) factorisations and apply the corresponding orthogonal transformations to a specified matrix.

PDTRMM2 routine performs a product of a general matrix with a triangular matrix storing the result in the triangular matrix, which must have enough space to store it (the result will not be triangular in general).

6. EXPERIMENTAL RESULTS

6.1 *Parallel platform*

The parallel platform used to test the routines is a cluster of PC's. Each node in the cluster is a biprocessor computer with 2 Pentium III processors at 866 MHz, 512 Mb of RAM and Redhat Linux operating system. Each node also has three network interfaces: two Fast Ethernet interfaces integrated in the motherboard and one Gigabit Ethernet card.

The implemented routines have been tested with up to 5 nodes. However, each node has been treated as a single processor machine. The biprocessor feature has not been exploited. This is done to avoid confusing communications through the network with communications via shared memory.

6.2 *Problem to solve*

The Lyapunov equation used to test the routines is generated in a way similar to that explained in (Penzl, 1996), but specialised for the standard case and generating a square matrix B (in the reference B was a transposed vector).

The matrix A of size $n \times n$, $n = 3q$, is generated as $A = W_n^{-1}\mathrm{diag}(A_1, \ldots, A_q)W_n$, with

$$A_i = \begin{pmatrix} s_i & 0 & 0 \\ 0 & t_i & t_i \\ 0 & -t_i & t_i \end{pmatrix}. \qquad (13)$$

W_n is a matrix of size $n \times n$ whose elements are all one except those of the main diagonal which are zero. The parameters s_i and t_i determine the eigenvalues of the generated matrix. They are chosen as $s_i = t_i = t^i$ (continuous case).

The matrix B of size $n \times n$ is generated as the symmetric positive definite matrix with elements $b_{ij} = n - |i - j|, 1 \leq i, j \leq n$.

Two different problems have been generated to test the parallel routines. Both of them are continuous cases and are generated with values 1.0 and 1.01 for parameter t.

6.3 *Timing results*

Parallel executions have been done for problems of size 2001 with $t = 1.0$, block size 24, and $t = 1.01$, block size 16. These block sizes have been the better ones in the execution of the main routine in a single node. Each of these cases has been executed both using the Fast Ethernet network and the Gigabit Ethernet network.

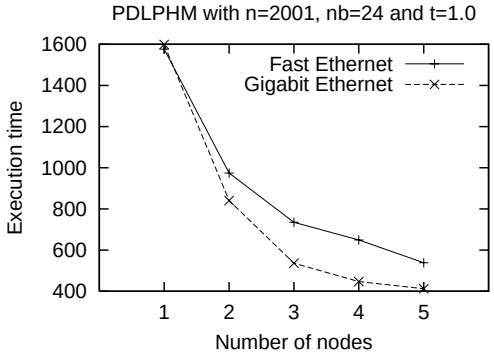

Fig. 1. Parallel execution times for t=1.0.

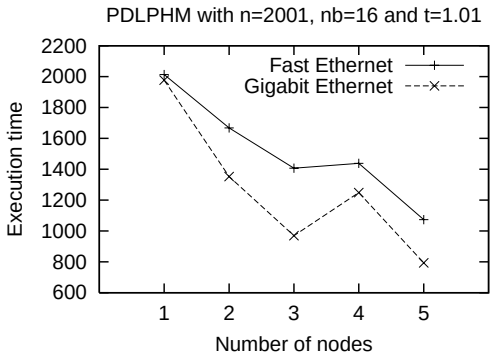

Fig. 2. Parallel execution times for t=1.01.

The times obtained (in seconds) can be seen in figures 1 and 2. As expected, the times using the Gigabit Ethernet network are lower than those of the Fast Ethernet network. However, differences are not as big as they may be expected. The reason for this can be that the developed algorithm does not have a large communications/computations

ratio. Computation is still the most costly part. The algorithm seems to use a sufficiently coarse grain, since a better network does not increase performance too much.

Table 1. Speedups and efficiencies.

Sp	P=1	P=2	P=3	P=4	P=5
t=1.0	1	1.9	2.98	3.58	3.88
t=1.01	1	1.46	2.04	1.58	2.49
E	P=1	P=2	P=3	P=4	P=5
t=1.0	1	0.95	0.99	0.90	0.78
t=1.01	1	0.73	0.68	0.40	0.50

Speedups and efficiencies are shown in table 1. The case with $t = 1.0$ has shown very good performance. Efficiencies of 90% up to 4 processors for a problem size of 2001 can be considered good results for this kind of problems (dense matrices).

However, the problem with $t = 1.01$ has much worse results. This implies that the cost of computing the Schur form depends most on the values of the initial problem, has better parallelism for a simpler matrix.

7. CONCLUSIONS

In this paper, the integration of routines to obtain a parallel algorithm to solve standard Lyapunov equations by Hammarling's method has been presented. It is a basic operation needed in the primary model reduction methods used in the SLICOT library. Thus, this is a first step to parallelise SLICOT model reduction routines, which is the main goal of Task II.C in the NICONET project.

Only a part of the necessary routines was implemented in the standard parallel library ScaLAPACK, thus implementations for the rest of the operations have been done. The main new routines are those in charge of the Schur factorisation (**PDGEES** routine is not completed in ScaLAPACK, yet the major part of its required routines are), the step of solving the reduced equation (explained in (Guerrero *et al.*, 2001)), and some other routines needed in the transformations (e.g. to compute the product of two matrices storing it in one of them).

Once all the necessary routines were available, the integration of calls among them has been implemented. The result is a main routine (**PDLPHM**) which allows to solve standard Lyapunov equations in parallel, both continuous and discrete versions, both in transpose or non-transpose form. This new routine and all the newly developed ones follow the same syntax and calling convention of ScaLAPACK, thus allowing any user of ScaLAPACK to work easily with them.

The parallel implementation has been tested with some problems in order to see the performance it can achieve. This performance has shown to be rather good. Only a part of the algorithm seems not to have good time results. This is the part in charge of the Schur factorisation, in particular the ScaLAPACK routine PDLAHQR. However, it may be due to the complexity to parallelise this operation, since the remainder of used ScaLAPACK routines work well.

As a parallel routine to solve standard Lyapunov equations is available, it is possible to deal with the parallelisation of the main SLICOT routines for the reduction of stable systems. This work is now being performed by the authors.

8. REFERENCES

Anderson, E., Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov and D. Sorenson (1992). *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics. Philadelphia, PA, USA.

Bartels, R. H. and G. W. Stewart (1972). Solution of the equation $AX + XB = C$. *Comm. ACM* **15**, 820–826.

Benner, Peter, Enrique S. Quintana-Orti and Gregorio Quintana-Orti (1999). A portable subroutine library for solving linear control problems on distributed memory computers. Technical Report NICONET 1999-1. Working Group on Software WGS. ESAT - Katholieke Universiteit Leuven (Belgium).

Guerrero, David, Vicente Hernández and José E. Román (2001). Parallel Solution of the Standard Lyapunov Equation by Hammarling's Method. Technical report. Third NICONET Workshop on Numerical Control Software. Louvain-la-Neuve (Belgium).

Guerrero, David, Vicente Hernández, José E. Román and Antonio M. Vidal (1998). Parallel Algorithms for the Cholesky Factor of Generalized Lyapunov Equations. In: *5th IFAC Workshop on Algorithms and Architectures for Real-Time Control*. Cancun, Mexico. pp. 237–242.

Hammarling, S. J. (1982). Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA J. of Numerical Analysis* **2**, 303–323.

Penzl, Thilo (1996). Numerical solution of generalized Lyapunov equations. Technical Report SFB393/96-02. Numerische Simulation auf massiv parallelen Rechnern. Technische Universität Chemnitz-Zwickau.

Varga, Andras (1999). Model Reduction Routines for SLICOT. Technical Report SLWN 1999-8. Working Group on Software WGS. ESAT - Katholieke Universiteit Leuven (Belgium).