Efficient Solution of Large Linear Systems in Model Reduction for VSLI Circuits

Alfredo Remón, Enrique S. Quintana-Ortí
Depto. de Ingeniería y Ciencia de Computadores, Universidad Jaume I
12.071-Castellón, Spain
{remon,quintana}@icc.uji.es

***Abstract*—We investigate the solution of linear systems that appear when model reduction via system balancing is applied in circuit simulation and design. In particular, we show how the properties and/or structure of the coefficient matrices allow the development and use of efficient (fast) parallel algorithms for the solution of the corresponding linear systems. Results are reported on a two-way Intel Xeon multiprocessor.**

Keywords—Large linear systems, model order reduction, system balancing, VLSI circuits, Lyapunov equations, LR-ADI iteration, multithreaded BLAS, multicore and SMP architectures.

I. Introduction

We consider linear dynamical systems, given in generalized state-space form by

$$\begin{array}{rcll} E\dot{x}(t) & = & Ax(t) + Bu(t), & t > 0, \\ y(t) & = & Cx(t) + Du(t), & t \geq 0, \end{array} \tag{1}$$

where $A, E \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{p\times m}$, and $x(0) = x^0 \in \mathbb{R}^n$ is the initial state. Here, n is the order of the system and the associated transfer function matrix (TFM) is $G(s) = C(sE - A)^{-1}B + D$. In model order reduction (MOR) we are interested in finding

$$\begin{array}{rcll} \hat{E}\dot{\hat{x}}(t) & = & \hat{A}\hat{x}(t) + \hat{B}u(t), & t > 0 \\ \hat{y}(t) & = & \hat{C}\hat{x}(t) + \hat{D}u(t), & t \geq 0, \end{array} \tag{2}$$

of order r, with $r \ll n$, and TFM $\hat{G}(s) = \hat{C}(s\hat{E} - \hat{A})^{-1}\hat{B} + \hat{D}$ which "approximates" $G(s)$.

Systems of the form (1) arise in circuit simulation and design (CSD) [1]. When modeling the interconnect or the pin package of VLSI circuits, the order is often too large to allow simulation in an adequate time or to even tackle the model using differential equation solvers. Therefore, MOR is frequently used to replace the circuit model by one of much smaller order. Here we will only consider methods based on system balancing, which are specially appealing in that they provide global computable error bounds and can preserve the system properties. For a survey of different MOR techniques, see [2].

There exist various methods for MOR which aim at balancing the system [2]. The core computation in all these is the solution of the Lyapunov equations

$$\begin{array}{rcl} A^T W_o E + E^T W_o A & = & BB^T, \\ A^T W_c E + E^T W_c A & = & C^T C, \end{array} \tag{3}$$

for Cholesky or full rank factors of $W_o, W_c \in \mathbb{R}^{n\times n}$. Lyapunov solvers based on the LR-ADI iteration [3] are specially efficient when both A and E are sparse, and the constant terms in (3) are of low numerical rank (a usual case in CSD). The LR-ADI method requires, at each iteration j, the solution of a linear system of the form

$$U_{j+1} := \left(\hat{A} + \gamma_j I_n\right)^{-1} U_j, \tag{4}$$

where and $\{\gamma_j\}_{j=0}^{\infty}$ are scalars with periodicity t_s, and $\{U_j\}_{j=0}^{\infty}$ have a small number of columns. Therefore, the linear systems in iterations j and $j + t_s$ share the same coefficient matrix and the use of direct solvers is highly recommendable. For further details on the LR-ADI iteration and its parallelization see, respectively, [3] and `http://www.pscom.uji.es/software.html`.

The coefficient matrices of the linear systems (4) associated with CSD models often present a moderate bandwidth (or can be transformed to that form), allowing the use of band solvers in LAPACK. Here we describe how specialized band solvers can be designed and utilized to efficiently exploit the properties and structure of these matrices. In particular, we propose a modified scheme for storing band matrices that yields a significant gain in the speed of the factorization stage, at the expense of a slightly larger work space. Also, for unsymmetric problems, the negative definiteness of the coefficient matrices may allow to obviate pivoting for stability during the factorization, resulting in higher performance; this then enables the introduction of BLAS-3 during the forward-substitution stage. Combined with a *multithreaded* implementation of BLAS, the codes allow the parallel solution of the linear systems on current multicore and SMP architectures. Preliminary experimental results on a two-way Intel Xeon multiprocessor provide evidence in support.

II. Efficient Solvers for Band Linear Systems

In this section we first describe the codes in LAPACK for the factorization of band matrices. We then propose a technique with the potential to attain higher efficiency. To illustrate this, we employ the routine in LAPACK for the Cholesky factorization of a band matrix, xpbtrf.

Given a s.p.d. matrix $\hat{A} \in \mathbb{R}^{n\times n}$ with bandwidth k_d, routine xpbtrf computes a lower (or upper) triangular factor $L \in \mathbb{R}^{n\times n}$, of bandwidth k_d, such that $\hat{A} = LL^T$. Upon completion, L overwrites the lower triangular part

of $\hat{A}$. Now, let $\hat{A}$ be partitioned as

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c|c|c|c|c} A_{00} & A_{01} & A_{02} & & \\ \hline A_{10} & A_{11} & A_{12} & A_{13} & \\ \hline A_{20} & A_{21} & A_{22} & A_{23} & A_{24} \\ \hline & A_{31} & A_{32} & A_{33} & A_{34} \\ \hline & & A_{42} & A_{43} & A_{44} \end{array}\right), \tag{5}$$

where $A_{TL}, A_{00} \in \mathbb{R}^{k\times k}$, $A_{11}, A_{33} \in \mathbb{R}^{n_b\times n_b}$, and $A_{22} \in \mathbb{R}^{k_l-n_b\times k_l-n_b}$. Here, the block size n_b is chosen for optimal performance. At a given iteration in xPBTRF, A_{TL} has been overwritten with the corresponding block of L, and A_{BL} and A_{BR} have been updated conformally. The following operations are then performed during the current iteration:

$$\begin{array}{lrcll} 1.1) & \text{Factorize } A_{11} & = & L_{11}L_{11}^T, & (\text{xPOTF2}) \\ 2.1) & A_{21}\ (=L_{21}) & := & A_{21}L_{11}^{-T}, & (\text{xTRSM}) \\ 2.2) & \text{TRIL}(A_{22}) & := & \text{TRIL}(A_{22}) - L_{21}L_{21}^T, & (\text{xSYRK}) \\ & W_k & := & \text{TRIU}(A_{31}), & \\ 3.1) & W_k & := & W_k L_{11}^{-T}, & (\text{xTRSM}) \\ 3.2) & A_{32} & := & A_{32} - W_k L_{21}^T, & (\text{xGEMM}) \\ 3.3) & \text{TRIL}(A_{33}) & := & \text{TRIL}(A_{33}) - W_k W_k^T, & (\text{xSYRK}) \\ & \text{TRIU}(A_{31})\ (=L_{31}) & := & W_k. & \end{array} \tag{6}$$

Here TRIU(A_{ij}) and TRIL(A_{ij}) denote, respectively, the upper and lower triangular part of A_{ij}, and the expressions are annotated with the LAPACK/BLAS routines that are employed for their computation. Provided $n_b \ll k_d$, a major part of the floating-point arithmetic operations (flops) are performed in terms of the BLAS-3 computation in 2.2), and high performance is to be expected if a tuned implementation of xSYRK is utilized. On the other hand, no attempt is made to exploit the upper triangular structure of A_{31}, L_{31} in 3.1)–3.3) as there is no appropriate kernel in BLAS. The packed storage scheme utilized for band matrices, and the use of BLAS kernels in 3.1)–3.3), results in the copies to/from the work space W_k.

The operations involving small blocks during the factorization stage in general do not attain high performance. While theoretically the influence of this part on the overall process should be small, practice has shown us otherwise. In some experiments with s.p.d. matrices, the optimal block size n_b determines that as much as 30–40% of the time is spent in these small operations; the ratio is even higher when multiple processors and a multithreaded BLAS are employed.

In order to overcome this problem, we propose to pad the data structure containing $\hat{A}$ with n_b rows of zeros in the bottom. In this way, the update of A_{31} can be combined with that of A_{21} in a single call to xTRSM, and a single call to xSYRK suffices to update A_{22}, A_{32}, and A_{33}. This strategy requires space for an extra $n_b \times n$ block which, provided $n_b \ll k_d$, is small. but no copies between/from W_k are then needed. The experimental results in the next section illustrate the practical benefits.

A similar strategy is applicable to the LU factorization code. Moreover, as preliminary experiments show, the coefficient matrices arising in CSD often do not require pivoting during the factorization so that, by utilizing a specially modification of LAPACK routine xGBTRF, higher performance can be expected. Also, in such a case BLAS-3 can be introduced in the forward substitution stage.

Example	DPBTRF (1 proc.)	DPBTRF (2 proc.)	DPBTRF+MS (2 proc.)
1	0.31	0.68	0.19
2	9.96	7.10	5.80
3	75.87	47.20	43.22

TABLE I

Execution time (sec.) of the factorization routines on the CSD examples.

III. Experimental Results

The following experiments were performed using IEEE double-precision (real) arithmetic on a parallel platform consisting of 2 Intel Xeon processors@2.4 GHz with 512 KB of L2 cache and 1 GB of RAM. The multithreaded BLAS implementation in GotoBLAS 1.00 was employed (`http://www.tacc.utexas.edu`). We consider three different examples from the Oberwolfach MOR benchmark collection (`http://www.imtek.uni-freiburg.de/simulation/benchmark`) corresponding to CSD models. The matrix $(E^{-1}A)$ in these systems is symmetric negative definite allowing the use of the Cholesky factorization during the factorization of the linear systems. MATLAB routine `symrcm` was used to reduce the bandwidth in Examples 2 and 3. A brief description of the examples follows:

Example 1. This is a model of a μthruster array with n=11445 states and bandwidth k_d=231.

Example 2. This model is used for 3D simulation of convective thermal flow in a chip with n=20082 and k_d=1226.

Example 3. This is a μmachined metal oxide gas sensor array with n=66917 and k_d=1957.

Although sparse (parallel) linear system solvers as SuperLU or MUMPS could be used, in some of these examples this leads to explosive fill-in so that memory is rapidly exhausted and the factorization is not possible and/or keeping the factors during the LR-ADI iteration becomes infeasible.

Table I reports the execution time required for the factorization stage of the matrices in the CSD examples. Algorithm xPBTRF refers to the original (double precision) code in LAPACK; DPBTRF+MS includes the modified storage scheme described at the end of the previous section. Using the two processors of the machine, the speed-up achieved by DPBTRF+MS w.r.t. DPBTRF is 3.51, 1.22, and 1.09. The high acceleration of the first case is due to the performance degradation that appears when multithreaded BLAS is used to factorize/update small-scale problems in DPBTRF (A_{11}, A_{21}, and A_{33}).

We expect similar improvements by applying the same technique to the solution of general band linear systems, which will surely be larger in case no pivoting is necessary.

References

[1] C.-K. Cheng, J. Lillis, S. Lin, and N.H. Chang, *Interconnect Analysis and Synthesis*, John Wiley & Sons, NY, 2000.

[2] A.C. Antoulas, *Lectures on the Approximation of Large-Scale Dynamical Systems*, SIAM Pub., Philadelphia, PA, 2005.

[3] T. Penzl, "A cyclic low rank Smith method for large sparse Lyapunov equations," *SIAM J. Sci. Comput.*, vol. 21, no. 4, pp. 1401–1418, 2000.