Balanced Truncation Model Order Reduction for LTI Systems with many Inputs or Outputs

Peter Benner and André Schneider

***Abstract*— We discuss balanced truncation (BT) based methods for model order reduction (MOR) of linear time invariant (LTI) systems with many input or many output terminals. Applying BT methods makes it necessary to balance the system, which is equivalent to finding the controllability and observability Gramian of the system in a special diagonal form. The Cholesky factors of these Gramians are efficiently computable as solutions of dual Lyapunov equations for systems with only few inputs and outputs. After a brief introduction and a short recollection of basic knowledge of BT, we show a method to get the Gramians' factors also for systems with many inputs and outputs with the help of the Gauss-Kronrod quadrature formula. We show some numerical results using this quadrature rule and explain how to get the BT reduced order model out of these results.**

I. Introduction

MOR turned out to be a powerful and necessary tool in the context of the simulation of various applications and problems over the last decades [1], [4], [18]. Modeling, especially in the area of circuit simulation but also in mechanical, biological and chemical applications, leads to LTI continuous-time systems of the form

$$\begin{aligned} \dot{x}(t) &= Ax(t)+Bu(t), \quad x(0)=x_0, \\ y(t) &= Cx(t)+Du(t). \end{aligned} \tag{1}$$

Here $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{p\times m}$, $x(t) \in \mathbb{R}^n$ contains internal state variables, $u(t) \in \mathbb{R}^m$ is the vector of input variables, $y(t) \in \mathbb{R}^p$ is the output vector, $x_0 \in \mathbb{R}^n$ is the initial value and n is the number of state variables, called the *order* of the system. So far, MOR methods concentrated on reducing the order of the system under the assumption that the dimensions of the input and output vector are much smaller than the order itself, i.e., $m,p \ll n$. Due to new applications this assumption is violated more and more often, e.g., when power grids are included in circuit simulation. Although there are methods trying to handle this challenge [8], [5], [13], the established methods – like BT methods [3], [11], [16], Krylov subspace methods [10], [9], proper orthogonal decomposition (POD) based methods, or reduced basis methods – have a common weakness in dealing with a lot of inputs or outputs. We explain a way to use balanced truncation methods efficiently in case of having either a lot of inputs or a lot of outputs.

II. Balanced Truncation

To apply balanced truncation we compute the controllability and observability Gramians

$$X = Y =: \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix} = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix}, \tag{2}$$

of (1) as solutions of the dual Lyapunov equations

$$AX + XA^T + BB^T = 0 \tag{3}$$

and

$$A^TY + YA + C^TC = 0. \tag{4}$$

Due to storage, efficiency and accuracy reasons, rather than computing X and Y themselves, usually one computes approximate factors X_C and Y_C of low rank such that

$$X \approx X_C X_C^T$$

and

$$Y \approx Y_C Y_C^T.$$

Using these factors, we compute a singular value decomposition of the form

$$X_C^T Y_C = [U_1 U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}. \tag{5}$$

Now we define the balancing transformations

$$T_l = Y_C V_1 \Sigma_1^{-1/2} \tag{6}$$

and

$$T_r = X_C U_1 \Sigma_1^{-1/2}, \tag{7}$$

where $\Sigma_1^{-1/2} = \operatorname{diag}(\frac{1}{\sqrt{\sigma_1}}, \ldots, \frac{1}{\sqrt{\sigma_k}})$, such that we are able to compute the reduced system as

$$(\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D}) := (T_l^T A T_r, T_l^T B, C T_r, D).$$

Note, that this (as an example) is the square root variant of balanced truncation. For the following we note that returning to the original formulation of Moore [15], T_l, T_r and the σ_i can also be computed from $X_C^T Y X_C$ or $Y_C^T X Y_C$, respectively.

P. Benner is with Technische Universität Chemnitz, Fakultät für Mathematik, 09107 Chemnitz, GERMANY and Max Planck Institute for Dynamics of Complex Technical Systems, 39106 Magdeburg, GERMANY benner@mpi-magdeburg.mpg.de

A. Schneider is with Technische Universität Chemnitz, Fakultät für Mathematik, 09107 Chemnitz, GERMANY andre.schneider@mathematik.tu-chemnitz.de

ISBN 978-963-311-370-7

III. Low Rank Alternating Direction Implicit (ADI) method

The efficient solution of the Lyapunov equations (3) and (4) is the bottleneck of BT MOR methods dealing with systems described above. Let us for now assume we look at systems (1) with many outputs, i.e., p is large and $m \ll n$. We consider (3)

$$AX + XA^T + BB^T = 0.$$

The ADI iteration for the Lyapunov equation is given by

$$\begin{array}{rcl} X_0 & = & 0, \\ (A + p_j I)X_{j-\frac{1}{2}} & = & -BB^T - X_{j-1}(A^T - p_j I), \\ (A + p_j I)X_j^T & = & -BB^T - X_{j-\frac{1}{2}}^T(A^T - p_j I), \end{array}$$

with $j = 1, \ldots, J$, see [21]. We rewrite this as a one step iteration

$$\begin{array}{rcl} X_0 & = & 0, \\ X_j & = & -2p_j(A + p_j I)^{-1}BB^T(A + p_j I)^{-T} \\ & & +(A + p_j I)^{-1}(A - p_j I)X_{j-1} \\ & & \cdot(A - p_j I)^T(A + p_j I)^{-T}, \end{array}$$

and insert the solution in factorized form, so we get

$$\begin{array}{rcl} X_{C,0}X_{C,0}^T & = & 0, \\ X_{C,j}X_{C,j}^T & = & -2p_j(A + p_j I)^{-1}BB^T(A + p_j I)^{-T} \\ & & +(A + p_j I)^{-1}(A - p_j I)X_{C,j-1} \\ & & \cdot X_{C,j-1}^T(A - p_j I)^T(A + p_j I)^{-T}. \end{array}$$

It is possible to express the low rank factor of the solution as

$$\begin{array}{rcl} X_{C,j} & = & \left[\sqrt{-2p_j}(A + p_j I)^{-1}B,\right. \\ & & \left.(A + p_j I)^{-1}(A - p_j I)X_{C,j-1}\right], \end{array}$$

but the number of columns to be processed grows in each step. Observing that $(A - p_i I)$, $(A + p_k I)^{-1}$ commute [12], we rewrite $X_{C,J}$ as

$$\begin{array}{rcl} X_{C,J} & = & \left[x_{C,J},\ P_{J-1}x_{C,J},\ P_{J-2}(P_{J-1}x_{C,J}),\ \ldots,\right. \\ & & \left.P_1(P_2 \cdots P_{J-1}x_{C,J})\right], \end{array}$$

where

$$x_{C,J} = \sqrt{-2p_J}(A + p_J I)^{-1}B$$

and

$$\begin{array}{rcl} P_i & := & \frac{\sqrt{-2p_i}}{\sqrt{-2p_{i+1}}}(A + p_i I)^{-1}(A - p_{i+1}I), \\ & = & \frac{\sqrt{-2p_i}}{\sqrt{-2p_{i+1}}}\left[I - (p_i + p_{i+1})(A + p_i I)^{-1}\right]. \end{array}$$

Using LRADI algorithms described in [17] we are able to compute X_C efficiently, but we fail doing the same with Y_C. Due to the fact that we add a fixed number of columns depending on the rank of C, which is at most p (and in fact often is), we end up with $Y_{C,J} \in \mathbb{R}^{n \times Jp}$, where Jp exceeds n easily for large p, and thus the LRADI iteration is not efficient anymore.

IV. The high rank solution

Equations (6) and (7) show that we need σ_i, $i = 1, \ldots, k$, for computing the reduced order model. Therefore again, we need X_C and Y_C. Because of the dimension of C also $\operatorname{rank}(Y_C)$ is large. It is too expensive to compute the factor of the solution of (4) and perform the SVD in (5). On the other hand, (2) shows that the nonzero σ_i are also computable as

$$\begin{array}{rcl} \sigma_i & = & \sqrt{\lambda_i(XY)} \\ & = & \sqrt{\lambda_i(X_C^T Y X_C)}, \end{array}$$

where $\lambda_i(\cdot)$ denotes the i-th eigenvalue of matrix $(\cdot)$, ordered by magnitude. From systems and control theory [20], we know that for $\omega \in \mathbb{R}$, we can express Y as

$$Y = \frac{1}{2\pi}\int_{-\infty}^{\infty}(j\omega I - A^T)^{-1}C^T C(j\omega I - A^T)^{-H}\,d\omega. \quad (8)$$

Using $X_C^T Y X_C$ and replacing Y as in (8), we get

$$\begin{array}{l} X_C^T Y X_C = \qquad (9) \\ \frac{1}{2\pi}\int_{-\infty}^{\infty}X_C^T(j\omega I - A^T)^{-1}C^T C(j\omega I - A^T)^{-H}X_C\,d\omega. \end{array}$$

With a suitable quadrature rule we can cheaply approximate the matrix product

$$X_C^T Y X_C \approx \frac{1}{2\pi}\sum_{l=0}^{N}\lambda_l f(\omega_l) \quad (10)$$

with

$$f(\omega) = X_C^T(j\omega I - A^T)^{-1}C^T C(j\omega I - A^T)^{-H}X_C,$$

if the evaluation of f is efficient enough, which is the case if X_C has just a few columns. Reason is the fact that computing $W(\omega) = (j\omega I - A^T)^{-H}X_C$ is equivalent to solving a sparse linear system of equations with few right-hand sides. Moreover, the function evaluation required to apply the quadrature rule then only needs matrix-vector multiplies of C^T or C with $W(\omega)^H$ or $W(\omega)$ (exploiting symmetry, only one of the products is necessary), respectively. This is usually very cheap, e.g., in circuit simulation often $C = [0,\ I_m]$ so that the matrix-vector multiplications come for free. We need an adaptive and highly accurate quadrature which, in the best case, provides an error estimation. We choose the Gauss-Kronrod quadrature formula explained in the next section as it allows approximating improper integrals and is capable to achieve a desired accuracy by adaptive refinement.

V. The Gauss-Kronrod quadrature formula

Well known is the n-point Gaussian quadrature rule for the integration over a domain $[a,b]$ (often $[-1,1]$)

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n}\lambda_i f(x_i)$$

with n suitable quadrature points x_i and weights λ_i. Gauss quadrature leads to exact results for polynomials of degree $2n-1$ or less. Gauss-Kronrod Quadrature is an enlargement of the n-point Gauss quadrature with $n+1$ new quadrature points and own weights a_i and b_i such that

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} a_i f(x_i) + \sum_{i=1}^{n+1} b_i f(y_i).$$

It is an example for a nested quadrature rule and leads to exact result for polynomials up to degree $3n+1$. The difference between the Gauss and the Gauss–Kronrod quadrature is often interpreted as error estimation of the integration. After an adaptive refinement of the integration domain $[a,b]$, a transformation of the quadrature points and weights per subdomain leads to an accurate approximation to the integral within the needed interval. For symmetric integrants this procedure simplifies. For more details of the computation of the Gauss–Kronrod quadrature, see [14], [6].

VI. Getting the reduced order model

From (10) we obtain an approximation to $X_C^T Y X_C$. We have to compute the projection matrices for balanced truncation to get the reduced order model. These projection matrices W and V are the result of a Schur decomposition of $X_C^T Y X_C$. [2] proposes this approach including a numerically efficient and accurate algorithm to compute the needed dominant invariant subspaces. First, a basis V_r of the right invariant subspace of the small size matrix $X_C^T Y X_C$ is computed by

$$(X_C^T Y X_C)V_r = V_r \Lambda_1,$$

where $\Lambda_1 = \mathrm{diag}(\lambda_1, \dots, \lambda_r)$, λ_i are the r largest eigenvalues of $X_C^T Y X_C$. The columns of V_r span the required subspace. Next, we calculate the left dominant invariant subspace basis W_l by

$$W_l^T (X_C^T Y X_C) = \Lambda_1 W_l^T.$$

Similarly to the balancing-free SR variant of BT we compute a QR decomposition of

$$V_r = Q_r R_r$$

and

$$W_l = Q_l R_l,$$

with the goal to orthogonalize V_r and W_l. Setting

$$T_r = Q_r$$

and

$$T_l = (Q_l^T Q_r)^{-1} Q_l^T,$$

we obtain the reduced order model by the projection

$$(\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D}) := (T_l^T A T_r, T_l^T B, C T_r, D).$$

VII. Numerical Investigations

From [7] we obtain a model of the Orr–Sommerfeld operator for the Taylor-Couette flow, which is in perturbation velocity variables

$$A = (-D^2)^{\frac{1}{2}} D^{-2} \left(-ijkD^2 + \frac{1}{Re} D^4 \right) (-D^2)^{-\frac{1}{2}}, \qquad (11)$$

with $D = \frac{d}{dy}$. We assume the inverse operators to be well defined. Equation (11) together with the definition of the input matrix $B \in \mathbb{R}^{100\times 5}$ as

$$B = \begin{bmatrix} I_5 \\ 0 \end{bmatrix},$$

and the output matrix $C \in \mathbb{R}^{80\times 100}$ as

$$C = \begin{bmatrix} I_{80} & 0 \end{bmatrix},$$

leads to system (1), which simulates the evolution of 2-dimensional perturbations in terms of velocities. Matrix A is of dimension 100×100 for Reynolds number $Re = 800$ and $k = 1$. We use an example of very small dimension to be able to compute both solution factors X_C and Y_C with the help of the LRADI method described above — a more efficient implementation and more involved numerical tests are work in progress and will be reported in the future.

We calculate the matrix product $(X_C^T Y X_C)_{ADI}$ and compare to the result of the approximation $(X_C^T Y X_C)_{GK}$ we obtain from (9) with help of the Gauss–Kronrod quadrature. The difference matrix $\|(X_C^T Y X_C)_{ADI} - (X_C^T Y X_C)_{GK}\|_F$ is analyzed in Frobenius norm.

Number of iterations integration domain refinement	$\|(X_C^T Y X_C)_{ADI} - (X_C^T Y X_C)_{GK}\|_F$
1	7.2836e-006
3	7.0056e-013
6	6.2575e-015

We see, the more we refine the domain of integration the more accurate the results are. Although we have no exact analysis of computational time the approach we propose is the method to choose in case of systems with many inputs or outputs.

VIII. Conclusions

If a method is available that computes X_C directly in an efficient way, e.g., Krylov subspace or ADI methods, see Sec. III and [16], [19], and X_C has few columns, then (9) can cheaply be approximated by a suitable quadrature rule. Gauss-Kronrod method fulfills the requirements of such a quadrature rule and also provides an estimation for the error caused by the numerical integration. Analog considerations can be done for the case of many inputs with just a few outputs. First numerical investigations using the Orr-Sommerfeld operator for Couette flow show that the method works accurately. In the future we will extend this work also for descriptor systems which opens the door for really large scale problems in the area of circuit simulation applications.

IX. ACKNOWLEDGMENTS

The work reported in this paper was supported by the German Federal Ministry of Education and Research (BMBF), grant no. 03BEPAE1. Responsibility for the contents of this publication rests with the authors.

REFERENCES

[1] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. SIAM Publications, Philadelphia, PA, 2005.

[2] U. Baur and P. Benner. Cross-Gramian based model reduction for data-sparse systems. *Electronic Transactions on Numerical Analysis*, 31:256–270, 2008.

[3] P. Benner. Advances in balancing-related model reduction for circuit simulation. In J. Roos and L. R. J. Costa, editors, *Scientific Computing in Electrical Engineering SCEE 2008*, volume 14 of *Mathematics in Industry*, pages 469–482. Springer-Verlag, Berlin/Heidelberg, Germany, 2010.

[4] P. Benner, V. Mehrmann, and D.C. Sorensen. *Dimension Reduction of Large-Scale Systems*, volume 45 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin/Heidelberg, Germany, 2005.

[5] P. Benner and A. Schneider. Model order and terminal reduction approaches via matrix decomposition and low rank approximation. In J. Roos and L. R. J. Costa, editors, *Scientific Computing in Electrical Engineering SCEE 2008*, volume 14 of *Mathematics in Industry*, pages 523–530. Springer-Verlag, Berlin/Heidelberg, Germany, 2010.

[6] D. Calvetti, G. H. Golub, W. B. Gragg, and L. Reichel. Computation of Gauss-Kronrod Quadrature Rules. *Math. Comp*, 69:1035–1052, 2000.

[7] Y. Chahlaoui and P. Van Dooren. A collection of benchmark examples for model reduction of linear time invariant dynamical systems, SLICOT Working Note 2002-2. 2002.

[8] P. Feldmann and F. Liu. Sparse and efficient reduced order modeling of linear subcircuits with large number of terminals. In *ICCAD '04: Proceedings of the 2004 IEEE/ACM Intl. Conf. Computer-aided design*, pages 88–92, Washington, DC, USA, 2004. IEEE Computer Society.

[9] R. W. Freund. On Padé-type model order reduction of J-Hermitian linear dynamical systems. *Linear Algebra and its Applications*, 429(10):2451 – 2464, 2008.

[10] Roland W. Freund. Passive reduced-order models for interconnect simulation and their computation via Krylov-subspace algorithms. In *DAC '99: Proceedings of the 36th annual ACM/IEEE Design Automation Conference*, pages 195–200, New York, NY, USA, 1999. ACM.

[11] J.-R. Li and J. White. Reduction of large circuit models via low rank approximate Gramians. *Int. J. Appl. Math. Comp. Sci.*, 11(5):1151–1171, 2001.

[12] J.-R. Li and J. White. Low rank solution of Lyapunov equations. *SIAM J. Matrix Anal. Appl.*, 24(1):260–280, 2002.

[13] P. Liu, S. X. D. Tan, B. Yan, and B. McGaughy. An efficient terminal and model order reduction algorithm. *Integr. VLSI J.*, 41(2):210–218, 2008.

[14] G. Monegato. An overview of the computational aspects of Kronrod quadrature rules. *Numer. Algorithms*, 26:173–196, 2001.

[15] B. C. Moore. Principal component analysis in linear systems: controllability, observability, and model reduction. *IEEE Trans. Automat. Control*, AC-26(1):17–32, 1981.

[16] T. Penzl. Algorithms for model reduction of large dynamical systems. *Linear Algebra Appl.*, 415(2–3):322–343, 2006. (Reprint of Technical Report SFB393/99-40, TU Chemnitz, 1999.).

[17] J. Saak. *Efficient Numerical Solution of Large Scale Algebraic Matrix Equations in PDE Control and Model Order Reduction*. PhD thesis, TU Chemnitz, July 2009. available from `http://www.tu-chemnitz.de/~saak/Data/Diss-web.pdf`.

[18] W.H.A. Schilders, H.A. van der Vorst, and J. Rommes. *Model Order Reduction: Theory, Research Aspects and Applications*. Springer-Verlag, Berlin, Heidelberg, 2008.

[19] V. Simoncini. A new iterative method for solving large-scale Lyapunov matrix equations. *SIAM J. Cont. Optim.*, 29(3):1268–1288, 2007.

[20] E.D. Sontag. *Mathematical Control Theory*. Springer-Verlag, New York, NY, 2nd edition, 1998.

[21] E.L. Wachspress. Iterative solution of the Lyapunov matrix equation. *Appl. Math. Letters*, 107:87–90, 1988.