

# PARALLEL ALGORITHMS FOR MODEL REDUCTION OF SPARSE SYSTEMS

José M. Badía<sup>1</sup>, Peter Benner<sup>2</sup>, Rafael Mayo<sup>1</sup>, Enrique S. Quintana-Orti<sup>1</sup>

<sup>1</sup>Depto. de Ingeniería y Ciencia de Computadores  
Universidad Jaume I de Castellón (Spain)  
{badia,mayo,quintana}@icc.uji.es

<sup>2</sup>Fakultät für Mathematik  
Technische Universität Chemnitz, Chemnitz (Germany)  
benner@mathematik.tu-chemnitz.de

PARA'04 - June 2004

## Linear Systems

Linear time-invariant systems:

$$\dot{x}(t) = Ax(t) + Bu(t), \quad t > 0, \quad x(0) = x^0,$$

$$y(t) = Cx(t) + Du(t), \quad t \geq 0,$$

- $n$  state-space variables, i.e.,  $n$  is the order of the system;
- $m$  inputs,
- $p$  outputs,
- $A$  is stable.

Corresponding TFM:

$$G(s) = C(sI_n - A)^{-1}B + D.$$



## Model Reduction: Purpose

Given

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t), & t > 0, & \quad x(0) = x^0, \\ y(t) &= Cx(t) + Du(t), & t \geq 0,\end{aligned}$$

find a **reduced model**

$$\begin{aligned}\dot{\hat{x}}(t) &= \hat{A}\hat{x}(t) + \hat{B}u(t), & t > 0, & \quad \hat{x}(0) = \hat{x}^0, \\ \hat{y}(t) &= \hat{C}\hat{x}(t) + \hat{D}u(t), & t \geq 0,\end{aligned}$$

of order  $r \ll n$  and output error

$$y - \hat{y} = Gu - \hat{G}u = (G - \hat{G})u$$

such that

$$\|y - \hat{y}\| \text{ and } \|G - \hat{G}\| \text{ is "small" !}$$



## Model Reduction: Motivation

### Control design:

- Real-time control is only possible with controllers of low complexity.
- The more complex the controller is, the more fragile.
- Control and optimization of systems governed by PDEs is impossible for large-scale systems arising from FE discretization.

### Simulation:

Repeated simulation for different force terms (input signals).

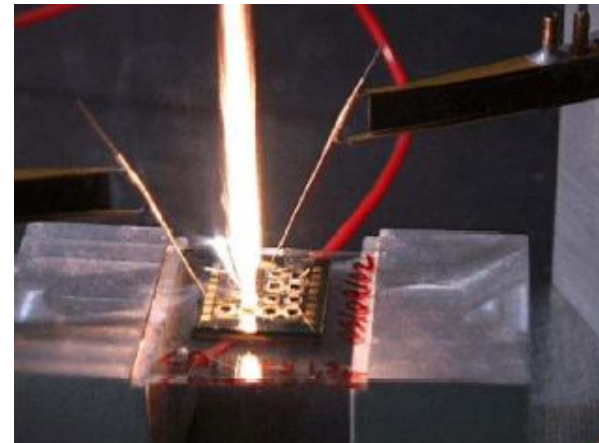
- VLSI chip design.
- Simulation of coupled PDE systems.
- Compact models for  $\mu$ -electro-mechanical systems (MEMS).



## Model Reduction: MEMS Example

$\mu$ -thruster array [IMTEK (UNIV. FREIBURG)/EU PROJECT  $\mu$ -PYROS]

- Co-integration of solid fuel with silicon  $\mu$ -machined system.
- Used for “nano-satellites” and gas generation.
- Design problem: reach the ignition temperature within the fuel without reaching the critical temperature at the neighbour  $\mu$ -thrusters (boundary).



$n$  from 4,257 – 79,171 states,  $p = 7$  outputs.

## Model Reduction: Other Examples

- $\mu$ -mechanical gyro for inertial simulation.  
 $n = 17,361$  states,  $m = 1$  input,  $p = 12$  outputs.
- Electrical networks.  
Order depends on the number of electrical devices in the interconnect.
- Optimal cooling of steel profiles.  
Order depending on the mesh discretization.



## Outline

1. Methods for model reduction: SRBT.
2. Solution of large-scale Lyapunov equations and other kernels.
3. Parallelization.
4. Numerical results.
5. Friendly access.
6. Conclusions.



## Methods for Model Reduction

(Antoulas'02):

- Krylov-based approximation methods.
  - Numerically efficient and applicable to large-scale (sparse) systems.
- SVD-based approximation methods.
  - Preserve of stability.
  - Provide a global error bound on  $\|G - \hat{G}\|$ .
  - Numerically efficient but **applicable to large-scale (sparse) systems?**



## Truncation Methods

Given a state-space transformation  $T \in \mathbb{R}^{n \times n}$ ,

$$TAT^{-1} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad TB = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \quad TC = (C_1, C_2),$$

with  $A_{11} \in \mathbb{R}^{r \times r}, \dots$

Partition  $T = [T_l^T W_l^T]^T$ ,  $T_l \in \mathbb{R}^{r \times n}$ ,  $T^{-1} = [T_r, W_r]$ ,  $T_r \in \mathbb{R}^{n \times r}$ .

Reduced-order model is then:  $(\hat{A}, \hat{B}, \hat{C}, \hat{D}) = (T_l A T_r, T_l B, C T_r, D)$ .

Find  $T$  and  $r$  such that  $\|y - \hat{y}\|$  is "small". In absolute error methods:

$$\min \|G - \hat{G}\|_{\infty}$$

as

$$\|y - \hat{y}\|_2 \leq \|G - \hat{G}\|_{\infty} \|u\|_2.$$



## Balanced Truncation (Moore, 81)

1. Solve the “coupled” Lyapunov matrix equations

$$AW_c + W_cA^T + BB^T = 0, \quad A^TW_o + W_oA + C^TC = 0,$$

for  $S, R$  such that  $W_c = S^TS, W_o = R^TR$ .

2. Compute

$$SR^T = U\Sigma V^T = [U_1 \ U_2] \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix},$$

with  $\Sigma_1 \in \mathbb{R}^{r \times r}, \Sigma_2 \in \mathbb{R}^{(n-r) \times (n-r)}$ .

The **Hankel singular values**,  $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$ , measure **how much a state is involved in energy transfer from a given input to a certain output!**



## Balanced Truncation (Cont.)

3. In the **square-root** method (Heath et al, 87; Tombs, Postlethwaite'87):

$$T_l = \Sigma_1^{-1/2} V_1^T R \quad \text{and} \quad T_r = S^T U_1 \Sigma_1^{-1/2},$$

$$\text{and } (\hat{A}, \hat{B}, \hat{C}, \hat{D}) = (T_l A T_r, T_l B, C T_r, D).$$

- Computable error bound:

$$\|G - \hat{G}\|_\infty \leq 2 \sum_{k=r+1}^n \sigma_k.$$

- Allows adaptive choice of  $r$ .



## Balanced Truncation (Cont. II)

Given  $(A, B, C, D)$  with  $A$  (sparse and) large, and  $m, p \ll n \dots$

How do we solve the previous numerical problems?

1. Coupled Lyapunov equations.
2. SVD of matrix product.
3. Application of the SR formulae to obtain the reduced-order model.



## 1. Solution of Coupled Lyapunov Equations

- Traditional methods reduce  $A$  to (real) Schur form and solve the resulting “triangular” equation (Bartels, Stewart, 72; Hammarling, 82):
  - Produce dense Cholesky factors of order  $n$ .
  - Do not exploit sparsity of  $A$   
→ applicable up to  $\mathcal{O}(n^3)$ .
- Sign function methods compute an spectral projector (Roberts, 71):
  - More reliable if  $S$  and  $R$  are numerically singular.
  - Reduced form is better conditioned.
  - Also more efficient as, usually,  $\text{rank}(S), \text{rank}(R) \ll n \dots$
  - Highly parallel but do not exploit sparsity of  $A$   
→ applicable up to  $\mathcal{O}(n^4)$ .



## 1. Solution of Coupled Lyapunov Equations (Cont. I)

LR-ADI iteration (Penzl, 98; Li, White, 99-02; Antoulas et al, 00-03):

$$\begin{aligned} V_0 &= (A + p_1 I_n)^{-1} B, & \hat{S}_0 &= \sqrt{-2 \operatorname{Re}(p_1)} V_0, \\ V_{k+1} &= V_k - \delta_k (A + p_{k+1} I_n)^{-1} V_k, & \hat{S}_{k+1} &= \begin{bmatrix} \hat{S}_k \\ \gamma_k V_{k+1} \end{bmatrix}, \end{aligned} \quad (1)$$

where  $\gamma_k = \sqrt{\operatorname{Re}(p_{k+1}) / \operatorname{Re}(p_k)}$ .

- $p = \{p_1, p_2, \dots, p_l\}$  are the “shifts”.
- Stop when contribution of  $\gamma_k V_{k+1}$  to  $\hat{S}_{k+1}$  is “small”.
- Super-linear convergence.
- After  $k$  iterations,  $W_c = S^T S \approx \hat{S}_k \hat{S}_k^T$ , with  $\hat{S}_k \in \mathbb{R}^{n \times (k \cdot m)}$ .



## 1. Solution of Coupled Lyapunov equations (Cont. II)

Implementation:

- Set  $p$  must be closed under complex conjugate.
- Shifts are computed by means of Arnoldi/Lanczos iterations.
- The LR-ADI iteration requires the solution of linear systems with coefficient matrix  $A$ .
- The iteration is applied cyclically:  $p_{k+l} = p_l$   
→ reuse  $P(A + p_k I_n) = LU$  if available.
- It may require complex arithmetic even if all data matrices are real.



## 2. SVD of Matrix Product

Replace the Cholesky factors by their low-rank approximations in

$$SR^T \approx \hat{S}_k^T \hat{R}_k = U\Sigma V^T.$$

Implementation:

- The product  $\hat{S}_k^T \hat{R}_k$  is of order  $(k \cdot m) \times (k \cdot p)$ ,  $m, p \ll n$ ,  
→ use dense LA methods.
- Accuracy can be enhanced by computing the SVD without computing explicitly the product, but it is difficult to parallelize.



### 3. Application of SRBT Formulae

Computation of the projection matrices

$$\begin{aligned}T_l &= \Sigma_1^{-1/2} V_1^T \hat{R}_k^T, \\T_r &= \hat{S}_k U_1 \Sigma_1^{-1/2},\end{aligned}$$

only requires dense LA methods.

Computation of the reduced-order matrices

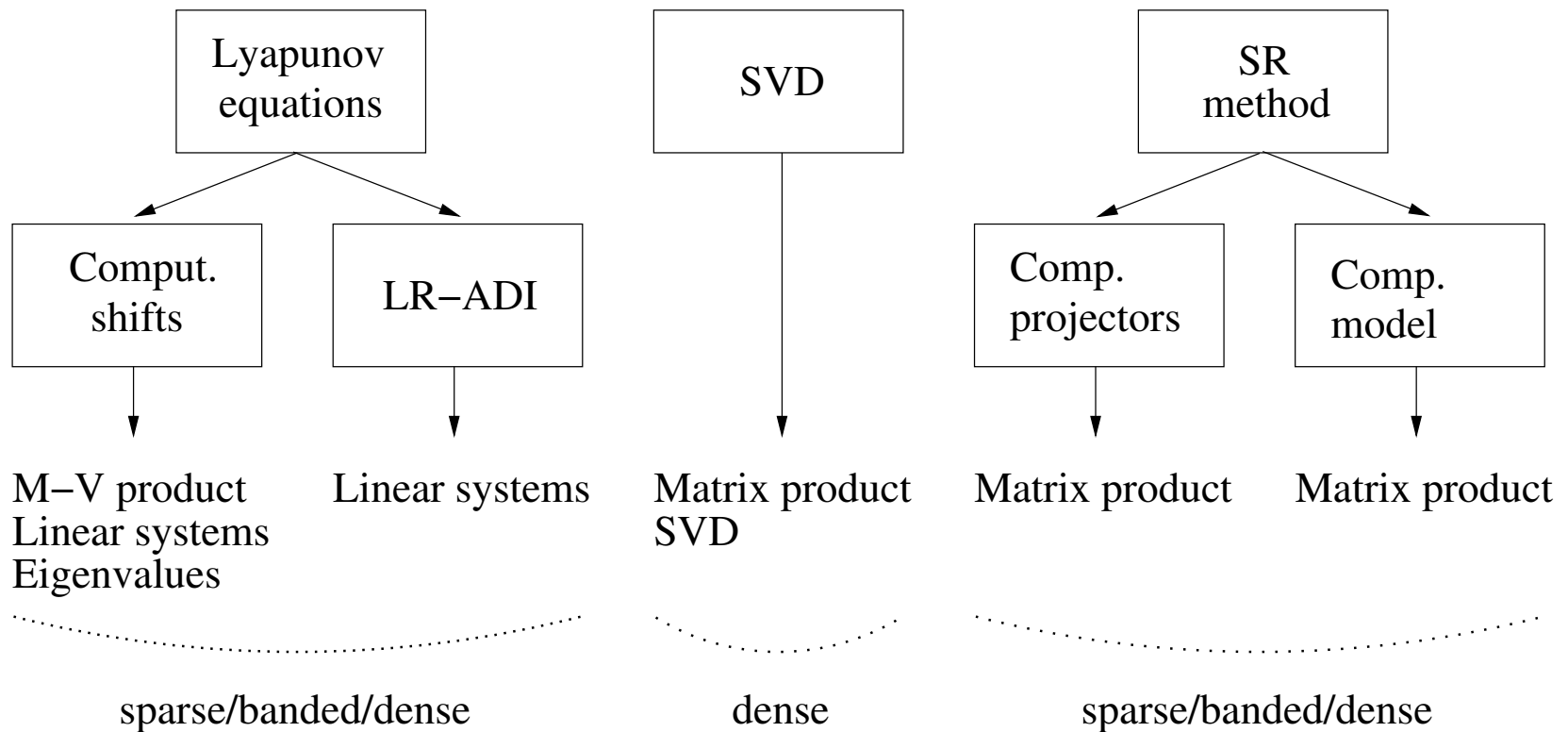
$$\begin{aligned}\hat{A} &= T_l A T_r, \\ \hat{B} &= T_l B, \\ \hat{C} &= C T_r,\end{aligned}$$

can be easily done exploiting any sparsity/pattern in  $A$ ,  $B$ , or  $C$ .



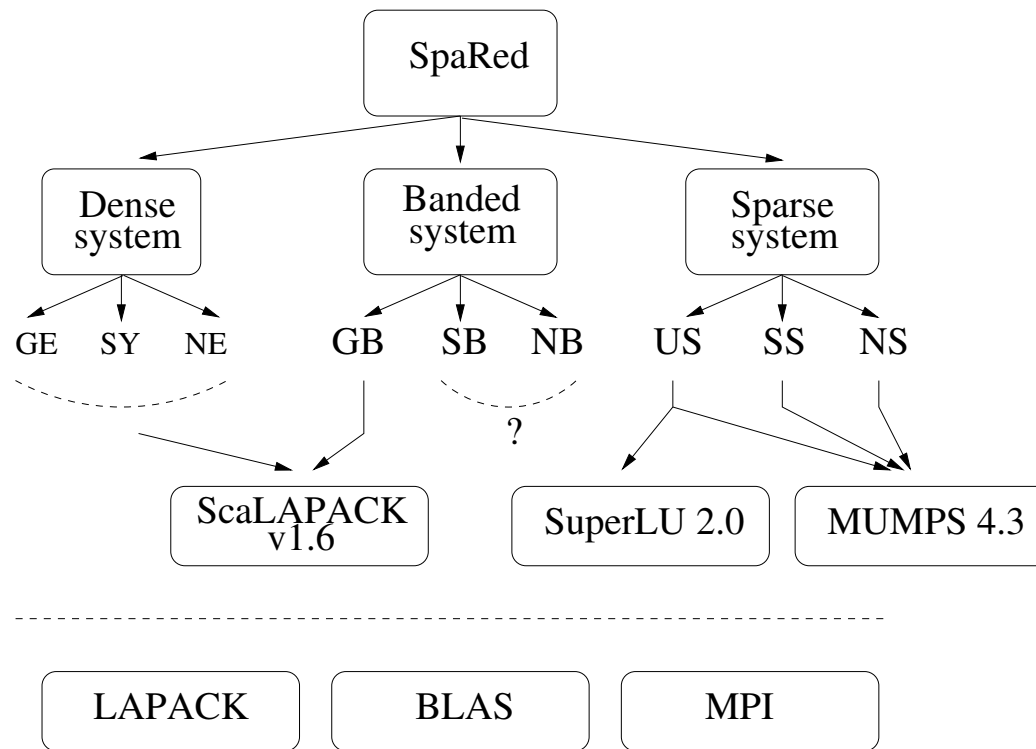
## Parallelization

Variety of LA operations:



## Parallelization (Cont.)

Use multiple parallel LA libraries:



## Experimental Results

Computing facility:

- 32 nodes  $\times$  2 Intel Pentium Xeon@2.4GHz, 1GB RAM connected via Myrinet interconnection switches, 2Gbps peak bandwidth.
- IEEE double precision arithmetic.



## Experimental Results (Cont. I)

Testbed:

1. Boundary control of heat flow in a thin rod.
2. 2-D heat diffusion in a square region.
3. RLC circuit.

This is not a comparison of performance of ScaLAPACK, SuperLU, and MUMPS banded/sparse linear system solvers!

To avoid it, we use a different example and a different problem size for each case.



## Experimental Results (Cont. II)

**Example 1.** From  $n = 10,000 \rightarrow r = 85; m = p = 1$ .

- SuperLU (distr.) 2.0 solver.
- $l = 50$  shifts.

Execution time using  $n_p = 16$  nodes:

Shifts	LR-ADI	SVD+SR	Total
11.74" (11.3%)	1' 28" (86.1%)	2.49" (2.4%)	1' 42"

- Convergence after 128 iterations  
 $\rightarrow$  both  $\hat{S}_k$  and  $\hat{R}_k$  of order  $n \times 128!$
- Absolute error  $\|G - \hat{G}\|_\infty \approx 1.91 \times 10^{-16}$ .



## Experimental Results (Cont. III)

**Example 2.** From  $n = 202,500 \rightarrow r = 30; m = p = 1$ .

- MUMPS 4.3.
- $l = 15$  shifts.

Execution time using  $n_p = 16$  nodes:

Shifts	LR-ADI	SVD+SR	Total
9.62" (0.7%)	19' 54" (97.6%)	19.9" (1.6%)	20' 22"

- Convergence after 69 iterations; both  $\hat{S}_k$  and  $\hat{R}_k$  of that order.
- Absolute error  $\|G - \hat{G}\|_\infty \approx 3.2 \times 10^{-17}$ .



## Experimental Results (Cont. IV)

**Example 3.** From  $n = 200,000 \rightarrow r = 50; m = p = 1$ .

- ScaLAPACK v1.6 (banded codes).
- $l = 15$  shifts.

Execution time using  $n_p = 16$  nodes:

Shifts	LR-ADI	SVD+SR	Total
2.26" (7.4%)	5.14" (16.9%)	22.87" (75.4%)	30.33"

- Convergence after 57 iterations,  $\hat{S}_k$  and  $\hat{R}_k$  of orders 57 and 50, resp.
- Absolute error  $\|G - \hat{G}\|_\infty < 4.9 \times 10^{-23}$ .



## Friendly Access (?)

Do you have a large-scale model to reduce and an appropriate cluster?

Steps:

1. Install BLAS, LAPACK, (and MPI?,)
2. Install SuperLU, MUMPS, ScaLAPACK,
3. Install our parallel model reduction codes,...



## Friendly Access (Cont.)

... or visit <http://spine.act.uji.es/~plicmr/SpaRedW3/SpaRedW3.html>

The screenshot shows a Netscape browser window titled "SpaRedW3: A Web Service for Model Reduction of Very Large-Scale Systems - Netscape". The address bar contains the URL "http://spine.act.uji.es/~plicmr/cgi-bin/SpaRedJobSub/index.php". The page content includes a logo of a network of nodes and the title "SpaRedW<sup>3</sup>: Job Submission Form".

1. <u>User identifier</u>	<input type="text"/>	2. <u>User password</u>	<input type="text"/>
3. <u>Model reduction method</u>	<input checked="" type="radio"/> Low Rank Square Root	4. <u>Solver</u>	<input checked="" type="radio"/> Default
5. <u>Order selection method</u>	<input checked="" type="radio"/> Fixed <input type="radio"/> Automatic		
6. <u>Number of states</u>	<input type="text"/>	7. <u>Number of inputs</u>	<input type="text"/>
8. <u>Number of outputs</u>	<input type="text"/>	9. <u>Order of reduced system</u>	<input type="text"/>
10. <u>Tolerance 1</u>	<input type="text"/>	11. <u>Tolerance 2</u>	<input type="text"/>
12. <u>Number of processors</u>	<input type="text"/>	13. <u>Compress tool</u>	<input checked="" type="radio"/> Not compressed <input type="radio"/> compress <input type="radio"/> gzip <input type="radio"/> zip
14. <u>Class of State Matrix</u>	<input checked="" type="radio"/> Dense <input type="radio"/> Symm. Band <input type="radio"/> General Band <input type="radio"/> General Sparse	15. <u>Class of Matrix Entries</u>	<input checked="" type="radio"/> Real <input type="radio"/> Complex
16. <u>e-mail</u>	<input type="text" value="yourmail@mail.server"/>		

File for A   File for B    
 File for C   File for D



## Concluding Remarks

- Existing libraries are not powerful enough:

SLICOT  $\rightarrow \mathcal{O}(10^3)$ .

PLiCMR  $\rightarrow \mathcal{O}(10^4)$  in parallel.

- Parallel SRBT algorithms in SpaRed allow reduction of sparse systems with  $\mathcal{O}(10^5)$  states.
- Efficacy depends on parallelism of underlying parallel libraries and, in the sparse case, in the sparsity pattern.
- Please, contact us if you have any large stable systems to reduce  
 $\rightarrow$  [quintana@icc.uji.es](mailto:quintana@icc.uji.es).

