Parallel Model Reduction of Large Dynamical Systems

Enrique S. Quintana-Ortí

Depto. de Ingeniería y Ciencia de Computadores
Universidad Jaume I de Castellón (Spain)
quintana@icc.uji.es

Joint work with:
José M. Badía (UJI), Peter Benner (TU-Chemnitz),
Rafael Mayo (UJI), and Gregorio Quintana-Ortí (UJI)

Austin'04 - September 2004

Dynamical Linear Systems

Linear time-invariant systems:

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

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

Corresponding TFM:

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

Large-scale for engineers means $n \approx 100,000 - 500,000$.

Goal for Model Reduction

Find a reduced-order model

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

of order $r \ll n$ such that the output error

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

is “small”.

Why?

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.

$\Longrightarrow$ a must!

Why (Cont.)?

Simulation:

Repeated simulation for different force terms (input signals).

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

$\Longrightarrow$ reduces the simulation time!

Example

μ-mechanical Gyroscope [The Imego Institute (Sweden) + Saab Bofors Dynamics AB]

- Commercial rate sensor with applications in inertial navigation systems.
- Simulation problem: Improve the design with respect to a number of parameters.
- $n = 17,361$ states.

Can we obtain a reduced-order model with similar behavior?

Outline

1. Truncation methods for model reduction.
2. Solution of Lyapunov equations.
3. Large problems: Parallelization.
4. Getting to the user.
5. Conclusions.

Outline

1. Truncation methods for model reduction.
   - Krylov-based methods.
   - SVD-based methods: Balanced Truncation.
2. Solution of Lyapunov equations.
3. Large problems: Parallelization.
4. Getting to the user.
5. Conclusions.

Rationale of Truncation Methods

Let

$$(A, B, C, D, x^0) \quad \text{and} \quad G(s) = C(sI_n - A)^{-1}B + D.$$

Consider a state-space transformation defined by $T \in \mathbb{R}^{n \times n}$ and

$$(\bar{A}, \bar{B}, \bar{C}, D, z^0) = (TAT^{-1}, TB, CT^{-1}, D, Tx^0).$$

Then,

$$\begin{aligned} \bar{G}(s) &= \bar{C}(sI_n - \bar{A})^{-1}\bar{B} + D \\ &= CT^{-1}(sI_n - TAT^{-1})^{-1}TB + D \\ &= C(sI_n - A)^{-1}B + D = G(s). \end{aligned}$$

Rationale of Truncation Methods (Cont.)

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

$$T = \begin{bmatrix} T_l \\ W_l \end{bmatrix} \quad \text{and} \quad T^{-1} = [T_r,\, W_r]\,,$$

with $T_l \in \mathbb{R}^{r \times n}$, $T_r \in \mathbb{R}^{n \times r}$.

Truncation methods compute the reduced-order model:

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

Goal: Find T_l and T_r and choose r such that $\|y - \hat{y}\|$ is “small”.

Taxonomy

(Antoulas'02):

- Krylov-based approximation methods.
  - Approach: Compute a low-dimensional subspace T that approximates the trajectory of $x(t)$ and project the system into that subspace.
  - Based on the Arnoldi iteration: composed of matrix-vector products.
  - Exploit/preserve sparsity.

$\Longrightarrow$ Applicable to large-scale (sparse) systems!

Taxonomy (Cont.)

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

Even for large systems the answer is yes! (provided we use parallel computing).

Balanced Truncation (Moore, 81)

One of many absolute error methods, which aim at

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

as

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

Here, $\|\cdot\|_\infty$ denotes the $\mathcal{H}_\infty$-norm which is ... too complex to define using words ;-)

Other methods: Hankel norm approximation, singular perturbation approximation, relative error methods, etc.

Balanced Truncation (Cont. I)

Composed of the following three steps:

Step 1. Solve the *coupled* Lyapunov matrix equations

$$\begin{aligned} AW_c + W_cA^T + BB^T &= 0, \\ A^TW_o + W_oA + C^TC &= 0, \end{aligned}$$

for the observability and controllability Gramians, W_c and W_o resp.

Actually, we need the Cholesky factors S and R such that

$$W_c = S^TS, \quad W_o = R^TR.$$

S and R are dense!

Balanced Truncation (Cont. II)

Step 2. Compute the Hankel singular values (HSV) from

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

with U, V, and Σ partitioned at a certain order r.

The HSV in $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$, measure how much a state is involved in energy transfer from a given input to a certain output!

Balanced Truncation (Cont. III)

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

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

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

- Computable error bound: $\boxed{\|G - \hat{G}\|_\infty \leq 2 \sum_{k=r+1}^{n} \sigma_k.}$
- Allows adaptive choice of r.

BT: Summary (in MATLAB)

```
>> % Step 1: Solve the coupled Lyapunov matrix equations
>> Wc = lyap(A,B*B');  S = chol(Wc);
>> Wo = lyap(A',C'*C); R = chol(Wo);
>>
>> % Step 2: Compute the HSV
>> [U,Sigma,V] = svd(S*R');
>>
>> % Step 3: Apply the SRBT method
>> U1 = U(:,1:r); V1 = V(:,1:r); Sigma1 = Sigma(1:r,1:r);
>>
>> T_l = inv(Sigma1.^(1/2)) * V1' * R;
>> T_r = S' * U1 * inv(Sigma1.^(1/2));
>>
>> Ar = T_l * A * T_r; Br = T_l * B; Cr = C * T_r;
```

Balanced Truncation (Cont. IV)

Given (A, B, C, D, x^0) with A large, and $m, p \ll n \ldots$

How do we solve the previous numerical problems?

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

Outline

1. Truncation methods for model reduction: SVD-based approach.
2. Solution of Lyapunov equations.
    - Traditional methods.
    - Sign function methods.
    - LR-ADI iteration.
3. Large problems: Parallelization.
4. Getting to the user.
5. Conclusions.

Case Study I

CD player.

- Tracking the lens of a CD player.
- Design problem: design low-cost controller that makes servo-system faster and robust to shocks.
- $n = 120$ states, $m = 2$ inputs, $p = 2$ outputs.

Traditional Methods (Bartels, Stewart, 72)

Consider the (real) Schur decomposition of A

$$A = U^T \breve{A} U,$$

where $\breve{A}$ is (quasi-)triangular and U is orthogonal. Then,

$$A^T W_o + W_o A + C^T C = 0 \Longrightarrow$$

$$U(A^T W_o + W_o A + C^T C = 0)U^T \Longrightarrow$$

$$U A^T U^T U W_o U^T + U W_o U^T U A U^T + U C^T C U^T = 0 \Longrightarrow$$

$$\breve{A}^T \breve{W}_o + \breve{W}_o \breve{A} + \breve{C}^T \breve{C} = 0 \quad \equiv$$

is a "reduced" form of this equation.

Traditional Methods (Cont. I)

Method:

1. Obtain the Schur decomposition of $A = U^T \check{A} U$.
2. Compute $\check{C} = C U^T$.
3. Solve the reduced equation

   $$\check{A}^T \check{W}_o + \check{W}_o \check{A} + \check{C}^T \check{C} = 0 \quad \equiv$$

   by "back-substitution".
4. Compute W_o from $W_o = U^T \check{W}_o U$.

A variation allows to obtain the Cholesky factor of $\check{W}_o$ and from there that of W_o (Hammarling, 82).

Experimental Results: CD Player

Model reduction using Bartels-Stewart method:

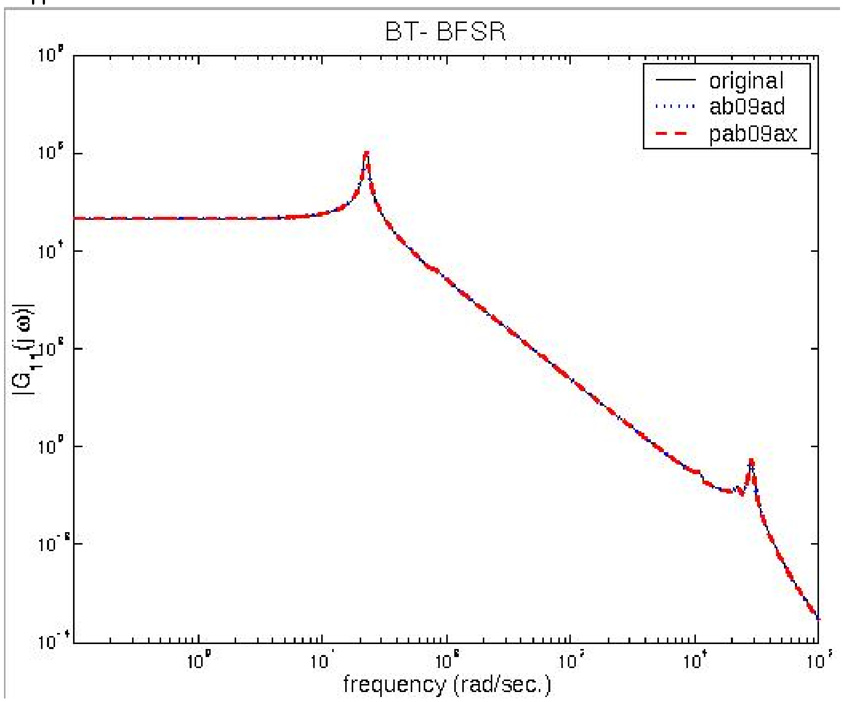
	CD player
n	120
r	42
n_p	1
Time	$0.68''$
$\|G - \hat{G}\|_\infty$	$1.6e-01$

- Less than $1''$!
- Allows construction of a cheaper controller!
- $\|G - \hat{G}\|_\infty \approx 1.6e-01$

Isn't that bad?

Experimental Results: CD Player

Remember: $\|G - \hat{G}\|_\infty$ is an absolute error!



Traditional Methods: Properties

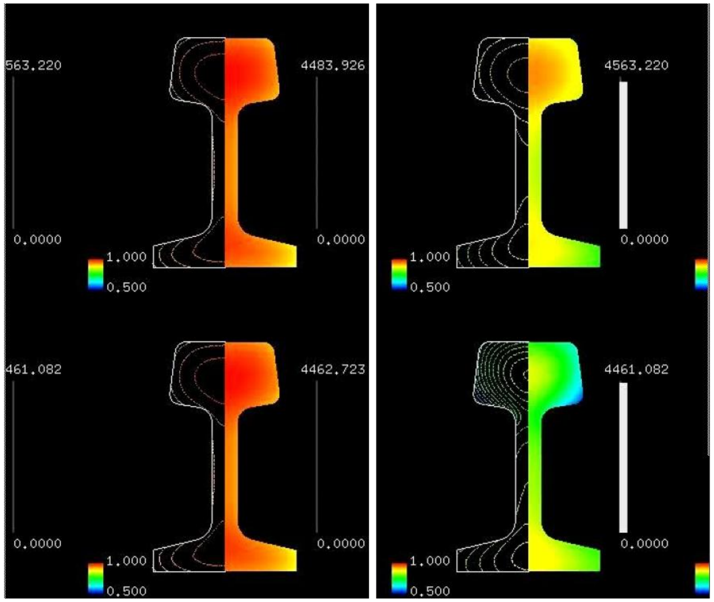
- Function `lyap` in MATLAB®.
- Currently based on routines of the same functionality in SLICOT (NICONET European Joint Project): http://win.tue.nl/niconet.
- The (real) Schur form is computed via the QR algorithm.
- Deliver Cholesky factors of order n.
- Do not exploit sparsity of A.
- Difficult to parallelize.

$\Longrightarrow$ applicable up to $\mathcal{O}(10^3)$.

Case Study II

Optimal cooling of steel profiles.

- Part of a manufacturing method for steel profiles.
- Design problem: design control that achieves moderate gradient temperatures when cooling from $1,000^o$ C to 500^o C.
- $n = 5,171$ states, $m = 7$ inputs, $p = 6$ outputs.



Sign Function Methods

Given $\alpha \in \mathbb{R}$,

$$\operatorname{sign}(\alpha) = \begin{cases} \quad 1 \text{ if } \alpha > 0, \\ -1 \text{ if } \alpha < 0, \\ \text{undefined otherwise.} \end{cases}$$

For a matrix $A \in \mathbb{R}^{n \times n}$, $\operatorname{sign}(A)$ is a function of the signs of its eigenvalues.

Given

$$H = \begin{bmatrix} A & 0 \\ C^T C & -A^T \end{bmatrix}, \quad \operatorname{sign}(H) = \begin{bmatrix} -I_n & 0 \\ 2W_o & I_n \end{bmatrix},$$

where W_o is the observability Gramian.

So, how do we compute the sign function?

Sign Function Methods (Cont. I)

For $H = \begin{bmatrix} A & 0 \\ C^T C & -A^T \end{bmatrix}$ the classical Newton iteration boils down to

$$\begin{aligned} A_{j+1} &= \frac{1}{2}(A_j + A_j^{-1})/2, \quad A_0 = A, \\ R_{j+1} &= \frac{1}{\sqrt{2}} \begin{bmatrix} R_j \\ R_j A_j^{-1} \end{bmatrix}, \quad R_0 = C, \end{aligned}$$

which converges to R, the Cholesky factor of W_o.

At each iteration R_j is increased in p rows (p being the number of outputs).

Sign Function Methods (Cont. II)

As in model reduction R (and S) is usually rank-deficient the cost of the iteration and subsequent steps can be greatly reduced (Benner, Quintana, 98):

At the jth iteration, compute the rank-revealing QR (RRQR) factorization

$$\frac{1}{\sqrt{2}}\left[\begin{array}{c} R_j \\ R_j A_j^{-1} \end{array}\right] = \bar{Q}\bar{R}\Pi$$

and then set

$$R_{j+1} = (\bar{R}\Pi)^T.$$

On convergence the iteration produces dense, full-rank $\hat{R}$ with $l \ll n$ columns, such that

$$\hat{R}^T \hat{R} \approx R^T R = W_o.$$

Implications on Steps 2 and 3

Replace the Cholesky factors by their (dense) low-rank approximations in

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

as the product $\hat{S}^T \hat{R}$ is of order $k \times l$, with k, $l \ll n$.

The computation of the projection matrices

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

is also cheaper.

Experimental Results

Cluster with 32 nodes $\times$ 2 Intel Pentium Xeon@2.4GHz, 1GB RAM, connected with Myrinet switches, 2Gbps peak bandwidth.

Experimental Results: Optimal cooling of steel profiles.

Parallel model reduction via sign function:

	Profiles
n	$5,177$
r	40
n_p	32
Time	$38'33''$
$\|G - \hat{G}\|_\infty$	$3.5e-04$

- Takes $\approx$ 40' to reduce Example 6 from order 5,177 to 40.
- Remember, the reduced-order model serves two purposes:
  - It is frequently necessary for control design.
  - Reduces simulation time.
- Reduce once, use it as many times as you want!

Sign Function Methods: Properties

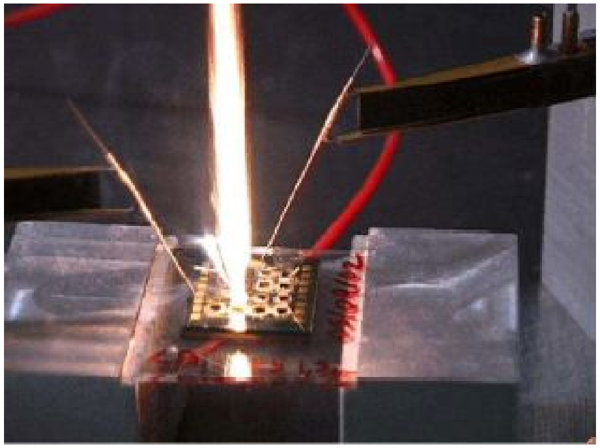
- More reliable if S and R are numerically singular.
- Reduced form is better conditioned.
- Also more efficient as usually $\mathrm{rank}\,(S)$, $\mathrm{rank}\,(R) \ll n\ldots$
- Ultimately, quadratic convergence.
- Highly parallel, as demonstrated in PLiCMR (Benner, Quintana-Ortí$\times 2$): `http://spine.act.uji.es/~plicmr`.
- Do not exploit any sparsity: The inverse of a sparse matrix is, in general, dense.

$\Longrightarrow$ applicable up to $\mathcal{O}(10^4)$.

Case Study III

μ-thruster array [IMTEK (Univ. Freiburg)/EU project μ-pyros]

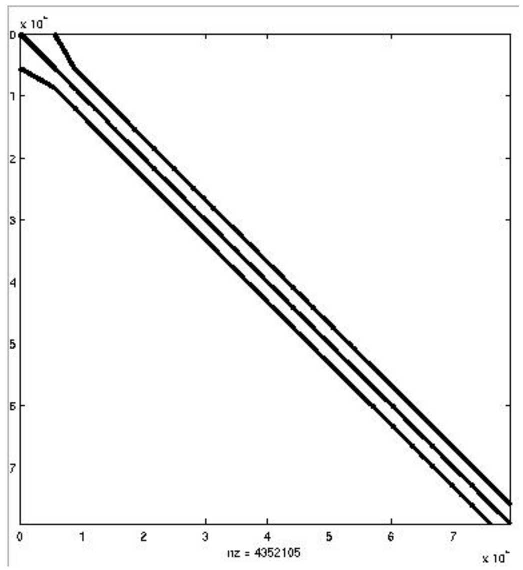
- Co-integration of solid fuel with silicon μ-machined system.
- Used for "nano-satellites" and gas generation.
- Design problem: reach the ignition temperature within the fuel without reaching the critical temperature at the neighbour μ-thrusters.
- n from $4{,}257 - 79{,}177$ states, $p = 7$ outputs.



Case Study III: μ-thruster array

Large-scale problems in model reduction are usually sparse.

State matrix:



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

Consider

$$AW_c + W_c A^T = BB^T.$$

The LR-ADI iteration is defined as:

$$\begin{array}{rclrcl} V_0 & = & (A + p_1 I_n)^{-1} B, & \hat{S}_0 & = & \sqrt{-2\ \mathrm{Re}(p_1)}\ V_0, \\ V_{j+1} & = & V_j - \delta_j (A + p_{j+1} I_n)^{-1} V_j, & \hat{S}_{j+1} & = & \left[\hat{S}_j\ ,\ \gamma_j V_{j+1} \right], \end{array} \qquad (1)$$

where $\gamma_j = \sqrt{\mathrm{Re}(p_{j+1})/\mathrm{Re}(p_j)}$.

Here, $p = \{p_1, p_2, \ldots, p_l\}$ are the "shifts".

After j iterations, we obtain a dense factor $\hat{S}_j \in \mathbb{R}^{n \times (j \cdot m)}$ such that

$$\hat{S}_j \hat{S}_j^T \approx S^T S = W_c.$$

Experimental Results: μ-thruster array

Parallel model reduction via LR-ADI iteration:

	μ-thruster
n	$79,841$
r	60
n_p	16
Time	$6'58''$

LR-ADI Iteration: Properties

Properties of the LR-ADI iteration:

- As reliable as the sign function.
- Also as efficient as usually $\mathrm{rank}\,(S)$, $\mathrm{rank}\,(R) \ll n \ldots$
- At most, superlinear convergence.
- Parallelism dictated by the sparsity of A and the solver; see SpaRed: (Badía, Benner, Quintana-Ortí, Mayo): `http://spine.act.uji.es/~plicmr/SpaRedW3/SpaRed.html`.
- Exploit the sparsity of A.

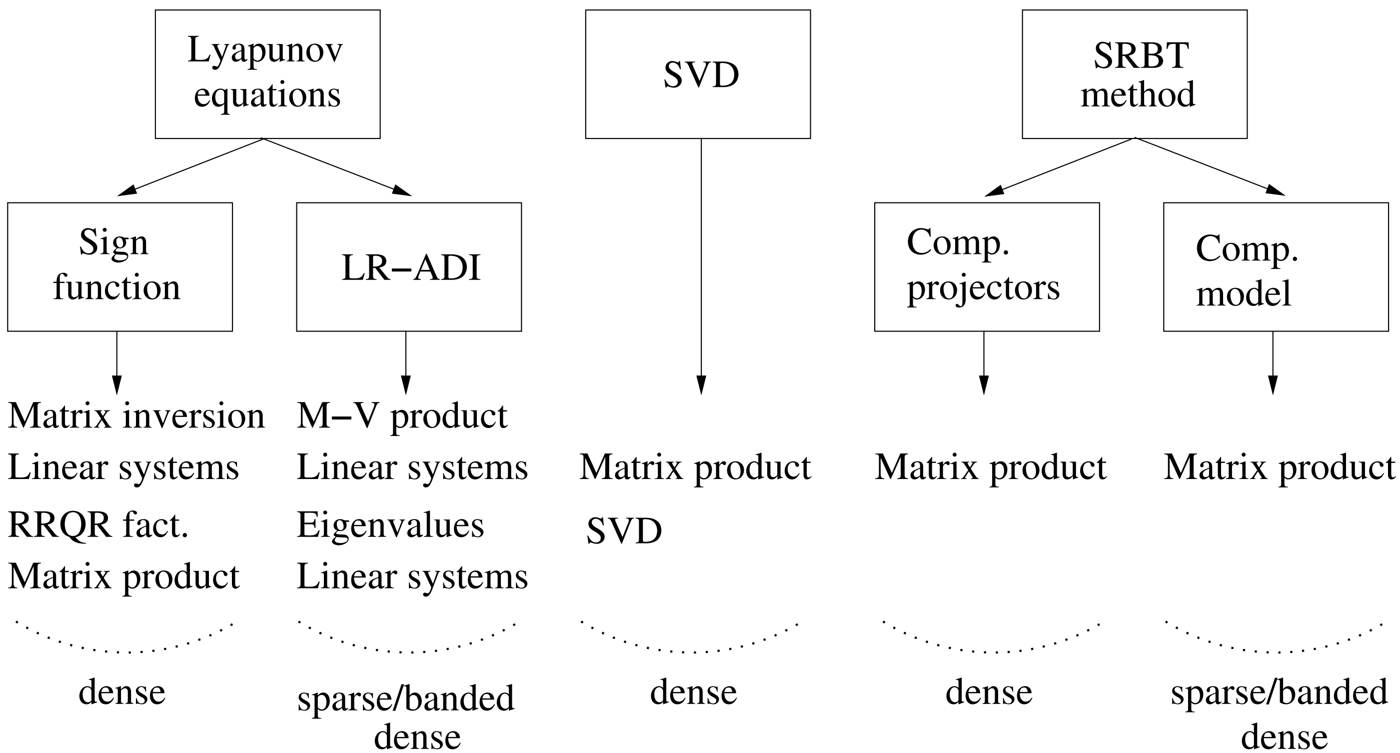
$\Longrightarrow$ applicable up to $\mathcal{O}(10^6)$, depending on the sparsity and solver.

Outline

1. Truncation methods for model reduction: SVD-based approach.
2. Solution of Lyapunov equations.
3. Large problems: Parallelization.
   - Use of parallel LA libraries.
4. Getting to the user.
5. Conclusions.

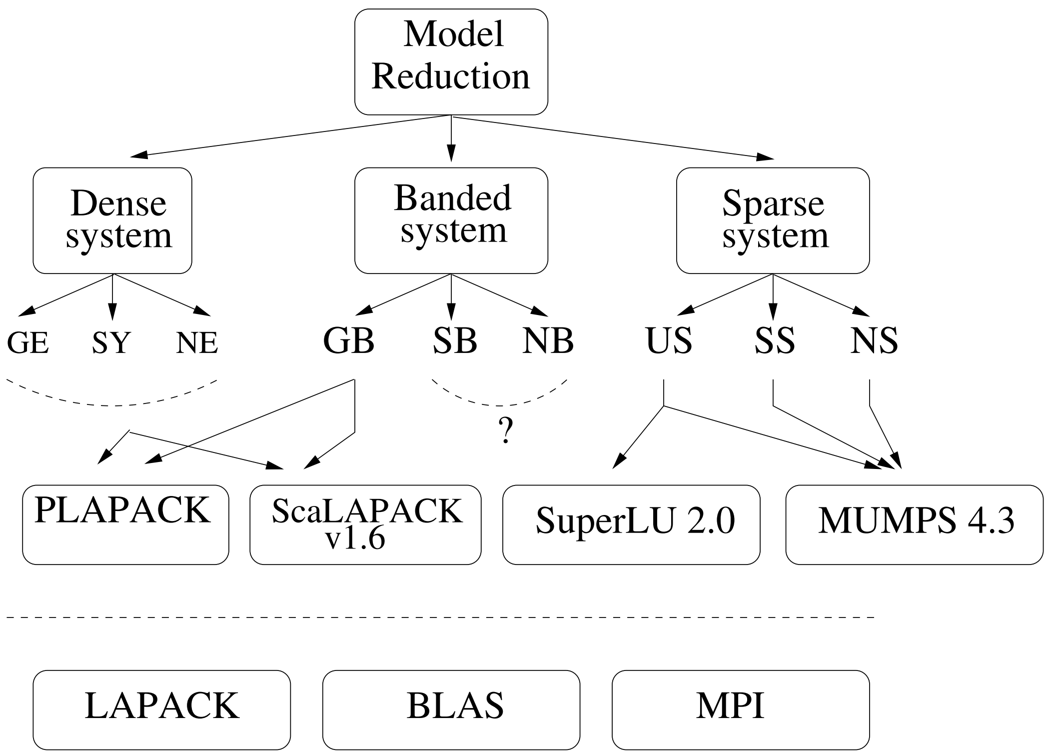
Parallelization

Variety of LA operations:



Parallelization (Cont.)

Use multiple parallel LA libraries:



Outline

1. Truncation methods for model reduction: SVD-based approach.
2. Solution of Lyapunov equations.
3. Large problems: Parallelization.
4. Getting to the user.
5. Conclusions.

Friendly Access (?)

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

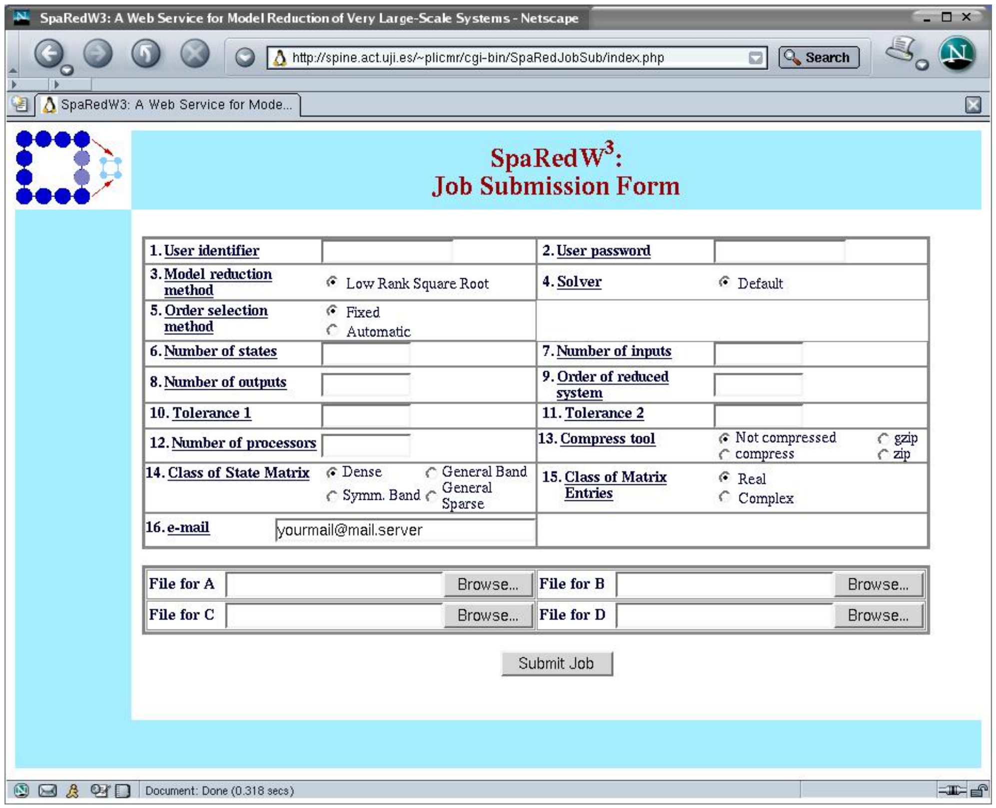
Steps:

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

Friendly Access (Cont.)

... or visit `http://spine.act.uji.es/~plicmr`

`http://spine.act.uji.es/~plicmr/SpaRedW3/SpaRedW3.html`



Concluding Remarks

- Krylov-based subspace methods are not enterily satisfactory.
- Existing serial libraries are not powerful enough:

  MATLAB/SLICOT $\rightarrow \mathcal{O}(10^3)$.
- Parallel model reduction algorithms in PLiCMR allow reduction of systems with $\mathcal{O}(10^4)$ states.
- Parallel SRBT algorithms in SpaRed allow reduction of sparse systems with $\mathcal{O}(10^6)$ 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 systems to reduce

  $\rightarrow$ `quintana@icc.uji.es`.