On a Parallel Implementation of the One-Sided Block Jacobi SVD Algorithm

Gabriel Okša [1], Martin Bečka, [1] Marián Vajteršic [2]

[1] Institute of Mathematics
Slovak Academy of Sciences
Bratislava, Slovakia

[2] Institute of Scientific Computing
University of Salzburg
Salzburg, Austria

Harrachov 2007, Czech Republic, August 19-25, 2007

Outline

Our task

Compute in parallel the Singular Value Decomposition (SVD) of a complex matrix A of the size $m \times n$, $m \geq n$:

$$A = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^H.,$$

where $U(m \times m)$ and $V(n \times n)$ are orthogonal and $\Sigma = \mathrm{diag}(\sigma_i)$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$.
Numerically stable way of computation:

- one- or two-sided block-Jacobi methods;
- large degree of parallelism.

Target architecture:

- distributed memory machines (parallel supercomputers and clusters) with Message Passing Interface (MPI).

Main structure of the OSBJA

- $A \in \mathbf{R}^{m \times n}$, $m \geq n$, its block-column partitioning $A = [A_1, A_2, \ldots, A_r]$, $n = (r-1)n_0 + n_r$, $n_r \leq n_0$.
- The OSBJA can be written as an iterative process:

$$\begin{aligned} A^{(0)} &= A, \quad V^{(0)} = I_n, \\ A^{(k+1)} &= A^{(k)} U^{(k)}, \quad V^{(k+1)} = V^{(k)} U^{(k)}, \quad k \geq 0, \end{aligned} \tag{1}$$

where the $n \times n$ OG matrix $U^{(k)}$ (block rotation):

$$U^{(k)} = \begin{pmatrix} I & & & & \\ & U_{ii}^{(k)} & & U_{ij}^{(k)} & \\ & & I & & \\ & U_{ji}^{(k)} & & U_{jj}^{(k)} & \\ & & & & I \end{pmatrix}. \tag{2}$$

- $A^{(k)} U^{(k)}$ in (1) mutually orthogonalizes the columns between $A_i^{(k)}$ and $A_j^{(k)}$.

Pivot strategy

- The orthogonal matrix

$$\hat{U}^{(k)} = \begin{pmatrix} U_{ii}^{(k)} & U_{ij}^{(k)} \\ U_{ji}^{(k)} & U_{jj}^{(k)} \end{pmatrix}$$

  of order $n_i + n_j$ is called the *pivot submatrix* of $U^{(k)}$ at step k.
- At each step k of the OSBJA, the pivot pair (i, j) is chosen according to a given pivot strategy $\mathcal{F} : \{0, 1, \ldots\} \to \mathbf{P}_r = \{(l, m) : 1 \leq l < m \leq r\}$.
- Well-known strategies: cyclic row, cyclic column.

3 parts of one (serial) step

- Part 1: The pivot pair (i, j) is given, compute the symmetric, positive semi-definite cross-product matrix

$$\hat{A}_{ij}^{(k)} = [A_i^{(k)} \; A_j^{(k)}]^T \, [A_i^{(k)} \; A_j^{(k)}] = \begin{pmatrix} A_i^{(k)T} A_i^{(k)} & A_i^{(k)T} A_j^{(k)} \\ A_j^{(k)T} A_i^{(k)} & A_j^{(k)T} A_j^{(k)} \end{pmatrix}.$$

  Complexity: $m\,(n_i + n_j)(n_i + n_j - 1)/2$ flops. When diagonal blocks of $\hat{A}_{ij}^{(k)}$ are diagonal, this reduces to $m\,(n_i\,n_j + n_i + n_j)$ flops.

- Part 2: $\hat{A}_{ij}^{(k)}$ is diagonalized by its eigenvalue decomposition:

$$\hat{U}^{(k)T} \, \hat{A}_{ij}^{(k)} \, \hat{U}^{(k)} = \hat{\Lambda}_{ij}^{(k)},$$

  Complexity: on average $8(n_i + n_j)^3$ flops (Hari 2005).

3 parts of one (serial) step (cont.)

- Part 3: $\hat{U}^{(k)}$ defines, after its partitioning, $U^{(k)}$ in (2), which is used for updates of $A^{(k)}$ and $V^{(k)}$ in (1).
  Complexity: $2m(n_i + n_j)^2$ flops.
- Overall complexity: One step of the OSBJA requires:

$$\begin{aligned} N_{\text{flop}}(k) &\approx m(n_i n_j + n_i + n_j) + 8(n_i + n_j)^3 + 2m(n_i + n_j)^2 \\ &= 64n_0^3 + (9n_0^2 + 2n_0)n \quad (\text{if } m = n = n_0 r) \end{aligned}$$

  flops.
- If we can choose the block width small enough, all computations of Part 2 can be performed in cache since $\hat{A}_{ij}^{(k)}$ is of order only $n_i + n_j$.
- The most complex computation is in Part 3 (matrix products). How to accelerate it?

Matrix preprocessing

- Step 1: QR factorization with column pivoting followed by the LQ factorization of the R-factor: $AP = Q_1 R$ and $R = LQ_2^T$. The OSBJA is then applied to the L-factor.
- Advantages:
  1. The off-diagonal norm of the L-factor is concentrated near the main diagonal and the Jacobi method using special orderings can be much faster and accurate (Hari 2005).
  2. Right singular vectors can be computed *a posteriori*: $LV = U\Sigma$, where $U\Sigma$ is the final stage of the Jacobi method applied to L (Drmač 1999). V is not updated during iterations!
- Then the SVD of A: $A = Q_1 L Q_2^T P^T = (Q_1 U)\Sigma(PQ_2 V)^T$ (postprocessing).
- Complexity: $2mn^2 + 1.5n^3$ flops + work in the Jacobi iterations.

Step 2: Special initialization of the recursion of three matrices.

We have: $L = [L_1, L_2, \ldots, L_r]$. Compute:

```
1. for $i = 1 : r$
2.     $\hat{A}_{ii}^{(0)} = L_i^T L_i$;
3.     $\hat{A}_{ii}^{(0)} = Q_{ii}^{(0)} \Gamma_i^{(0)} Q_{ii}^{(0)T}$; (spectral decomposition)
4.     $(A_i^{(0)} = L_i Q_{ii}^{(0)})$; (not performed, just the connection)
5. end;
```

Initialize three block matrices with blocks of small orders n_i, n_j:

$$\begin{aligned} B^{(0)} &= [B_1^{(0)}, B_2^{(0)}, \ldots, B_r^{(0)}] = [L_1, L_2, \ldots, L_r], \\ Q^{(0)} &= \mathrm{diag}(Q_{11}^{(0)}, Q_{22}^{(0)}, \ldots, Q_{rr}^{(0)}), \\ \Gamma^{(0)} &= \mathrm{diag}(\Gamma_1^{(0)}, \Gamma_2^{(0)}, \ldots, \Gamma_r^{(0)}) \quad \text{(stored as vector)}. \end{aligned}$$

Complexity: for $n = n_0 r$ about $(n/4 + 8n_0)n_0 n$ flops (assuming 4 sweeps in step 3).

Recursion $k \rightarrow k+1$

1. Compute the cross-product matrix:

$$\hat{A}_{ij}^{(k)} = \begin{pmatrix} Q_{ii}^{(k)} & \\ & Q_{jj}^{(k)} \end{pmatrix}^T \begin{pmatrix} B_i^{(k)T} B_i^{(k)} & B_i^{(k)T} B_j^{(k)} \\ B_j^{(k)T} B_i^{(k)} & B_j^{(k)T} B_j^{(k)} \end{pmatrix} \begin{pmatrix} Q_{ii}^{(k)} & \\ & Q_{jj}^{(k)} \end{pmatrix}$$

$$= \begin{pmatrix} \Gamma_i^{(k)} & \tilde{A}_{ij}^{(k)} \\ \tilde{A}_{ij}^{(k)T} & \Gamma_j^{(k)} \end{pmatrix}, \text{ where } \tilde{A}_{ij}^{(k)} \equiv Q_{ii}^{(k)T} (B_i^{(k)T} B_j^{(k)}) Q_{jj}^{(k)}.$$

2. Compute the eigendecomposition: $\hat{A}_{ij}^{(k)} = \hat{U}^{(k)} \hat{\Lambda}_{ij}^{(k)} \hat{U}^{(k)T}$.
Copy the eigenvalues into appropriate positions of $\Gamma^{(k+1)}$.

3. Compute the CS decomposition:

$$\hat{U}^{(k)} = \begin{pmatrix} V_{ii}^{(k)} & \\ & V_{jj}^{(k)} \end{pmatrix} \begin{pmatrix} C_{ii}^{(k)} & -S_{ij}^{(k)} \\ S_{ji}^{(k)} & C_{jj}^{(k)} \end{pmatrix} \begin{pmatrix} W_{ii}^{(k)} & \\ & W_{jj}^{(k)} \end{pmatrix}^T$$

$$\equiv \hat{V}^{(k)} \hat{T}^{(k)} \hat{W}^{(k)T}.$$

Recursion $k \to k+1$ (cont.)

4. Compute new matrices B and Q:

$$(B_i^{(k+1)}, B_j^{(k+1)}) = (B_i^{(k)}(Q_{ii}^{(k)} V_{ii}^{(k)}), B_j^{(k)}(Q_{jj}^{(k)} V_{jj}^{(k)})\, \hat{T}^{(k)},$$
$$Q_{ii}^{(k+1)} = W_{ii}^{(k)T}, \quad Q_{jj}^{(k+1)} = W_{jj}^{(k)T}.$$

Main idea: the 'large dimension' m is totally eliminated from recursion.

Matrix multiplications are of the form XY, where X is of order $n \times n_i$ or $n \times n_j$, and Y is square of order n_i or n_j.

Final update of B_i and B_j: special structure of $\hat{T}^{(k)}$ (rotations of columns of length n).

Overall complexity (Hari 2005):

$$N_{\text{flop}}(k) \approx 82n_0^3 + (3n_0^2 + 2n_0)n.$$

Stopping criterion

Let:

$$\hat{A}^{(k)} = A^{(k)T} A^{(k)} = \begin{pmatrix} \tilde{A}_{11}^{(k)} & \cdots & \tilde{A}_{1r}^{(k)} \\ \vdots & & \vdots \\ \tilde{A}_{1r}^{(k)T} & \cdots & \tilde{A}_{rr}^{(k)} \end{pmatrix},$$

$$D_k = (\mathrm{diag}(\hat{A}^{(k)}))^{1/2} = \mathrm{diag}(\|A^{(k)}e_1\|, \ldots, \|A^{(k)}e_n\|) = (\Gamma^{(k)})^{1/2}$$

$$\hat{A}_S^{(k)} = D_k^{-1} \hat{A}^{(k)} D_k^{-1} \text{ (scaled matrix).}$$

Two measures of convergence:

$$\alpha_k \equiv \mathrm{off}(\hat{A}_S^{(k)}) = \frac{\sqrt{2}}{2} \|\hat{A}_S^{(k)} - I\|_{\mathrm{F}},$$

$$\omega_k \equiv \mathrm{off}(\hat{A}^{(k)}) = \sqrt{\sum_{j=1}^{r} \sum_{t=j+1}^{r} \|\tilde{A}_{jt}^{(k)}\|_{\mathrm{F}}^2}.$$

Computation of α_k: $\approx n^3/2$ flops.

Stopping criterion (cont.)

Stopping criterion: $\alpha_k \leq n^2 \epsilon$.

At the end of block sweep $t+1$:

$$\omega^2_{(t+1)N} = \omega^2_{tN} - \sum_{k=tN}^{(t+1)N-1} \|\tilde{A}^{(k)}_{i(k),j(k)}\|^2_{\mathrm{F}}, \quad N = r(r-1)/2.$$

Easy update of ω^2, but a severe cancellation may occur.

Assuming the quadratic convergence of ω for $t \geq t_0$, monitor:

$$\nu_{tN} \equiv \sqrt{\sum_{k=tN}^{(t+1)N-1} \|\tilde{A}^{(k)}_{i(k),j(k)}\|^2_{\mathrm{F}}} = \omega_{tN} + O(\omega^2_{tN}), \quad t \geq t_0.$$

When $\nu_{tN} \leq n\sqrt{\gamma_1^{((t+1)N)}\epsilon}$ with $\gamma_1^{((t+1)N)} = \max[\Gamma^{((t+1)N)}]$, compute $\alpha_{(t+1)N}$.

Parallel implementation

- Data layout: p processors, two block columns per processor: $r = 2p$.
- Preprocessing: The QRFCP and the LQF via the ScaLAPACK's routine `PDGEQPF` and `PDGELQF`, respectively. Each processor computes its local cross-product matrix and two spectral decompositions of its diagonal blocks $\hat{A}_{ll}^{(0)}$.
- Recursion:
  1. Choose some parallel block ordering (e.g., round-robin).
  2. Each processor stores two block columns and matrix blocks B_i, B_j, Q_{ii}, Q_{jj}, and two vectors γ_i, γ_j. In the worst case, this amount of data must be transferred between processors at the beginning of each parallel iteration step ($= r/2$ subtasks) according to the ordering.
  3. All computations are local (LAPACK), no global communication.

Parallel implementation (cont.)

Stopping criterion: Its implementation requires the global communication.

- Update of ω^2 and ν: Local computation of the squared Frobenius norm of each nullified matrix block in each processor, then the global sum of local squares in PE 1, and, finally, the broadcast of an updated value to all processors: routines `MPI_ALLREDUCE`, `MPI_ALLGATHER` and `MPI_BCAST` from the ScaLAPACK.
- Computation of α: Scaling of the columns and rows of B by the values stored in vector γ, the square of Frobenius norm, the broadcast of new α: routines `MPI_ALLGATHERV`, `MPI_ALLREDUCE`, `MPI_ALLGATHER`, `MPI_BCAST`.

Conclusions

- Matrix preprocessing: Concentration of the Frobenius norm towards main diagonal leads to a substantial reduction of the number Jacobi sweeps needed for the convergence.
- Working with matrix blocks: Enables to accelerate the computation even in the serial case using a fast cache memory.
- Three-matrix recursion: Eliminates the ‘large’ dimension m, all updates can be made in a fast cache memory.
- Parallelization: Additional speed-up possible, especially for large n (number of columns).