The implicit Hari–Zimmermann algorithm for the generalized SVD

Sanja Singer[1] Vedran Novaković[2] Saša Singer[3]

[1]University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture, Croatia

[2]STFC, Daresbury Laboratory, United Kingdom

[3]University of Zagreb, Faculty of Science, Department of Mathematics, Croatia

Seminar za numeričku matematiku i znanstveno računanje,
14. 01. 2016., PMF-Matematički odsjek

This work has been supported in part by Croatian Science Foundation under the project IP–2014–09–3670.

Outline of the talk:

- brief description of the original Falk–Langemeyer algorithm, and the Hari–Zimmermann (HZ) algorithm for the GEP,
- description of the HZ algorithm for the GSVD computation,
- some implementation details,
- results of numerical testing.

The Falk–Langemeyer method

- invented in 1960, paper published in two parts, in the collection Elektronische Datenverarbeitung,
- quadratic convergence of the cyclic method is proved in M.Sc. thesis of Slapničar (1989, supervised by Hari),
- the method solves the Generalized Eigenvalue Problem (GEP) for a symmetric and definite matrix pair (A, B),
- it constructs a sequence of congruent pairs,

$$A^{(\ell+1)} = C_\ell^T A^{(\ell)} C_\ell, \quad B^{(\ell+1)} = C_\ell^T B^{(\ell)} C_\ell,$$

where $(A^{(1)}, B^{(1)}) := (A, B)$.

Symmetry is not enough (Parlett)

Example 1

$$A = B = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad \text{eigenpairs } (1, \mathrm{e}_1), \ \left(\frac{0}{0}, \mathrm{e}_2\right)$$

Example 2

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad \text{eigenpairs } \left(\frac{1}{0}, \mathrm{e}_1\right), \left(\frac{0}{1}, \mathrm{e}_2\right)$$

Example 3

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad \text{eigenpairs } \left(i, \begin{bmatrix} i \\ -1 \end{bmatrix}\right), \left(i, \begin{bmatrix} i \\ 1 \end{bmatrix}\right)$$

The Hari–Zimmermann method

- Zimmermann in her Ph.D. thesis (1969) briefly sketched a method for the GEP when B is positive definite,
- Hari in his Ph.D. thesis (1984) filled in the missing details, proved global and quadratic convergence (cyclic strategies)
- before the iterative part, the pair is scaled so that the diagonal elements of B are all equal to one,

$$A^{(1)} := DAD, \quad B^{(1)} := DBD,$$
$$D = \operatorname{diag}\left((b_{11})^{-1/2}, (b_{22})^{-1/2}, \ldots, (b_{kk})^{-1/2}\right),$$

- the method constructs a sequence of congruent pairs

$$A^{(\ell+1)} = Z_\ell^T A^{(\ell)} Z_\ell, \quad B^{(\ell+1)} = Z_\ell^T B^{(\ell)} Z_\ell.$$

The Hari–Zimmermann method for the GEP

The transformation matrix Z_ℓ

- resembles an ordinary plane rotation: it is the identity matrix, except for its (i,j)-restriction $\widehat{Z}_\ell$, where

$$\widehat{Z}_\ell = \frac{1}{\sqrt{1-\left(b_{ij}^{(\ell)}\right)^2}} \begin{bmatrix} \cos\varphi_\ell & \sin\varphi_\ell \\ -\sin\psi_\ell & \cos\psi_\ell \end{bmatrix},$$

- φ_ℓ and ψ_ℓ are determined so that the transformations diagonalize the pivot submatrices $\widehat{A}^{(\ell)}$ and $\widehat{B}^{(\ell)}$
- the transformation keeps the diagonal elements of B intact
- if $B = I$ then Z_ℓ is the ordinary rotation, the method is the ordinary Jacobi method for a single matrix.

The Hari–Zimmermann method for the GEP

Computation of the elements of $\widehat{Z}_\ell$

- for simplicity, the transformation index ℓ is omitted

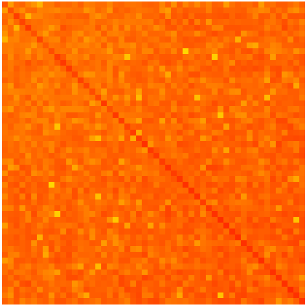
$$\tan(2\vartheta) = \frac{2a_{ij} - (a_{ii} + a_{jj})b_{ij}}{(a_{jj} - a_{ii})\sqrt{1 - (b_{ij})^2}}, \qquad -\frac{\pi}{4} < \vartheta \leq \frac{\pi}{4}$$

$$\xi = \frac{b_{ij}}{\sqrt{1 + b_{ij}} + \sqrt{1 - b_{ij}}}$$

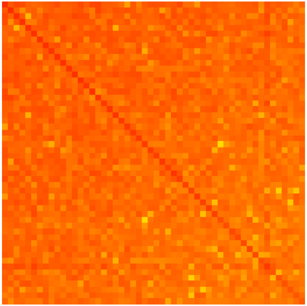
$$\eta = \frac{b_{ij}}{\left(1 + \sqrt{1 + b_{ij}}\right)\left(1 + \sqrt{1 - b_{ij}}\right)}$$

$$\begin{aligned}
\cos\varphi &= \cos\vartheta + \xi(\sin\vartheta - \eta\cos\vartheta) \\
\cos\psi &= \cos\vartheta - \xi(\sin\vartheta + \eta\cos\vartheta) \\
\sin\varphi &= \sin\vartheta - \xi(\cos\vartheta + \eta\sin\vartheta) \\
\sin\psi &= \sin\vartheta + \xi(\cos\vartheta - \eta\sin\vartheta)
\end{aligned}$$

An example — A and B positive definite of order 52



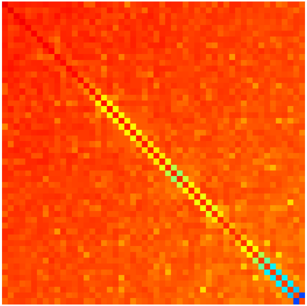
A



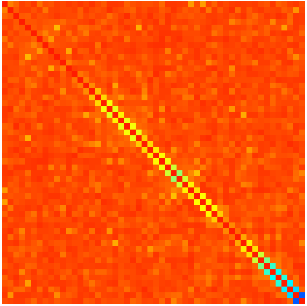
B

the starting pair

An example — A and B positive definite of order 52



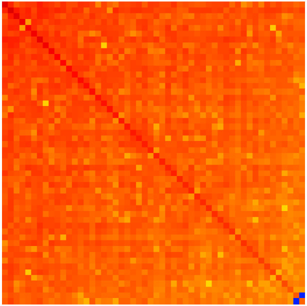
A



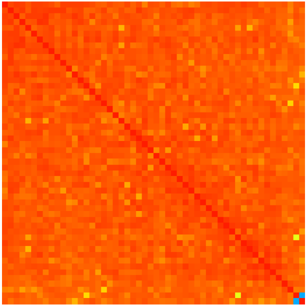
B

end of sweep 1

An example — A and B positive definite of order 52



A

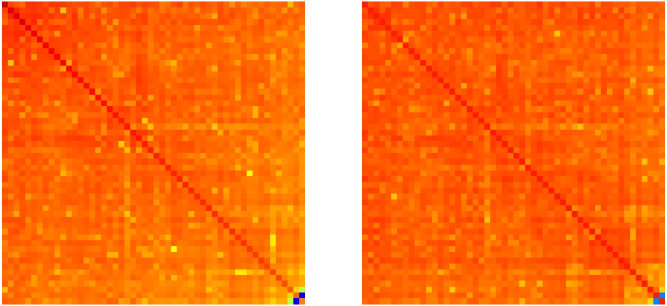


B

end of sweep 2

The pointwise algorithm for the GEP

An example — A and B positive definite of order 52

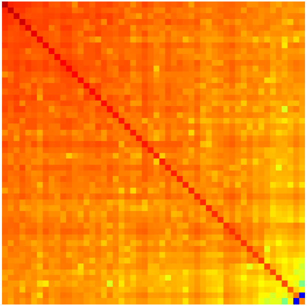


A $\qquad$ B

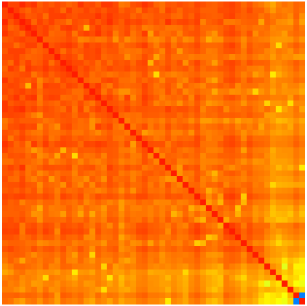
end of sweep 3

The pointwise algorithm for the GEP

An example — A and B positive definite of order 52



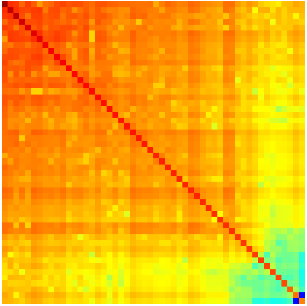
A



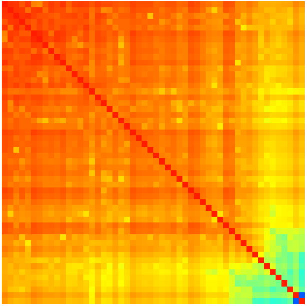
B

end of sweep 4

An example — A and B positive definite of order 52



A

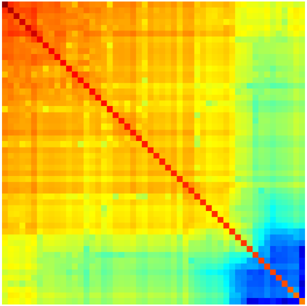


B

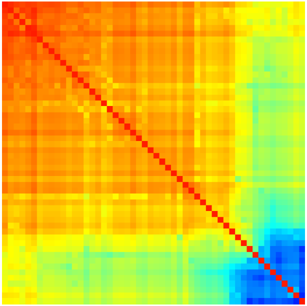
end of sweep 5

The pointwise algorithm for the GEP

An example — A and B positive definite of order 52



A

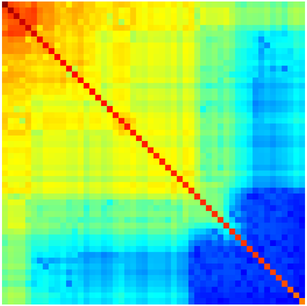


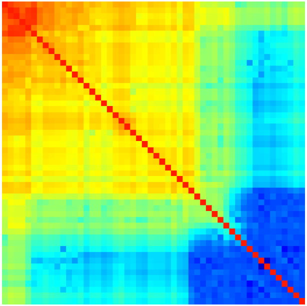
B

end of sweep 6

The pointwise algorithm for the GEP

An example — A and B positive definite of order 52

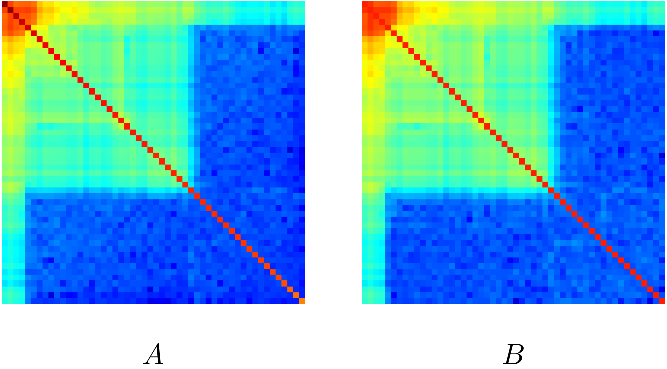




A B

end of sweep 7

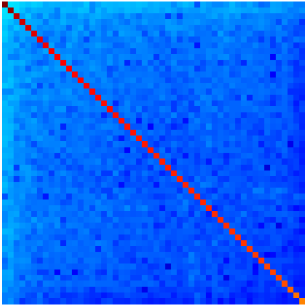
An example — A and B positive definite of order 52



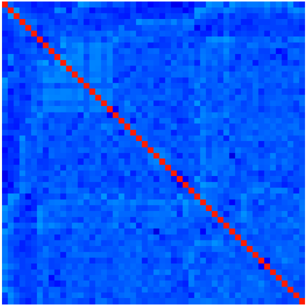
end of sweep 8

The pointwise algorithm for the GEP

An example — A and B positive definite of order 52



A



B

end of sweep 9

The generalized SVD

Definition

- For given matrices $F \in \mathbb{C}^{m \times n}$ and $G \in \mathbb{C}^{p \times n}$, where

$$K = \begin{bmatrix} F \\ G \end{bmatrix}, \quad k = \operatorname{rank}(K),$$

  there exist unitary matrices $U \in \mathbb{C}^{m \times m}$, $V \in \mathbb{C}^{p \times p}$, and a matrix $X \in \mathbb{C}^{k \times n}$, such that

$$F = U \Sigma_F X, \quad G = V \Sigma_G X, \quad \Sigma_F \in \mathbb{R}^{m \times k}, \quad \Sigma_G \in \mathbb{R}^{p \times k}.$$

- Σ_F and Σ_G are real, "diagonal", and nonnegative.
- Furthermore, Σ_F and Σ_G satisfy

$$\Sigma_F^T \Sigma_F + \Sigma_G^T \Sigma_G = I.$$

- The ratios $(\Sigma_F)_{ii} / (\Sigma_G)_{ii}$ are called the generalized singular values of the pair (F, G).

Connection between the GEP and the GSVD

- Given matrices: $F_0 \in \mathbb{R}^{m\times n}$ and $G_0 \in \mathbb{R}^{p\times n}$.
- If G_0 is not of full column rank, then use, for example, LAPACK preprocessing to obtain square matrices (F, G), with G of full rank k.
- For such F and G, since G^TG is a positive definite matrix, the pair (F^TF, G^TG) in the corresponding GEP is symmetric and definite.
- There exist many nonsingular matrices Z that simultaneously diagonalize (F^TF, G^TG) by congruences,

$$Z^TF^TFZ = \Lambda_F, \quad Z^TG^TGZ = \Lambda_G,$$

where Λ_F and Λ_G are diagonal, $(\Lambda_F)_{ii} \geq 0$ and $(\Lambda_G)_{ii} > 0$, for $i = 1, \ldots, k$.

Connection between the GEP and the GSVD

- Since Λ_F and Λ_G are diagonal, the columns of FZ and GZ are orthogonal (not orthonormal),

$$FZ = U\Lambda_F^{1/2}, \quad GZ = V\Lambda_G^{1/2},$$

  where U and V are orthogonal matrices.

- If $\Lambda_F + \Lambda_G \neq I$, then the matrices in the GSVD are

$$X := SZ^{-1}, \qquad \Sigma_F := \Lambda_F^{1/2}S^{-1}, \qquad \Sigma_G := \Lambda_G^{1/2}S^{-1}.$$

  where $S = (\Lambda_F + \Lambda_G)^{1/2}$ is the diagonal scaling.

- If only the generalized singular values are needed, rescaling is not necessary, and $\sigma_i = (\Lambda_G^{-1/2}\Lambda_F^{1/2})_{ii}$, for $i = 1, \ldots, k$.

The pointwise algorithm for the GSVD

The implicit HZ algorithm for the GSVD

$Z = I; \qquad it = 0$

`repeat` // sweep loop

$it = it + 1$

`for` all pairs (i, j), $1 \le i < j \le k$

compute

$$\widehat{A} = \begin{bmatrix} f_i^T f_i & f_i^T f_j \\ f_i^T f_j & f_j^T f_j \end{bmatrix}; \qquad \widehat{B} = \begin{bmatrix} g_i^T g_i & g_i^T g_j \\ g_i^T g_j & g_j^T g_j \end{bmatrix}$$

compute the elements of $\widehat{Z}$

// transform F, G and Z

$[f_i, f_j] = [f_i, f_j] \cdot \widehat{Z}$

$[g_i, g_j] = [g_i, g_j] \cdot \widehat{Z}$

$[z_i, z_j] = [z_i, z_j] \cdot \widehat{Z}$

`until` (no transf. in this sweep) `or` $(it \ge maxcyc)$)

How to make the algorithm faster and more accurate

Sequential algorithms

- blocking – each block has $k_i \approx k/nb$ columns

$$F = [F_1, F_2, \ldots, F_{nb}], \quad G = [G_1, G_2, \ldots, G_{nb}].$$

- each pivot block can either be fully orthogonalized – full-block algorithm, or,
- each pair of columns in each block is orthogonalized once in a sweep – block oriented algorithm
- pivoting – transformations are applied in such way that after each transformation it holds

$$\frac{\|f_i'\|_2}{\|g_i'\|_2} \geq \frac{\|f_j'\|_2}{\|g_j'\|_2}, \quad i < j.$$

Numerical testing of the sequential algorithms

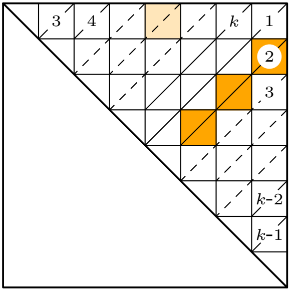
- Implementation: Fortran routines with MKL.

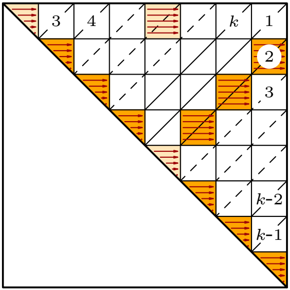
	with threaded MKL (12 cores)			
k	`DTGSJA`	`pointwise HZ`	`HZ-FB-32`	`HZ-BO-32`
500	16.16	3.17	4.36	2.03
1000	128.56	26.89	18.50	7.65
1500	466.11	105.31	42.38	19.31
2000	1092.39	273.48	86.01	41.60
2500	2186.39	547.84	139.53	73.07
3000	3726.76	1652.14	203.00	109.46
3500	6062.03	2480.14	294.58	186.40
4000	8976.99	3568.00	411.71	239.89
4500	12805.27	4910.09	553.67	343.58
5000	20110.39	6599.68	711.86	426.76

Times (in seconds).

Parallel pivoting strategy

- Choose pivot blocks independently in each step, for example, by using (block)-modulus strategy (not optimal!)





- stopping criterion
  - skip a transformation if cosines are 1
  - final stop — all transformations are skipped.

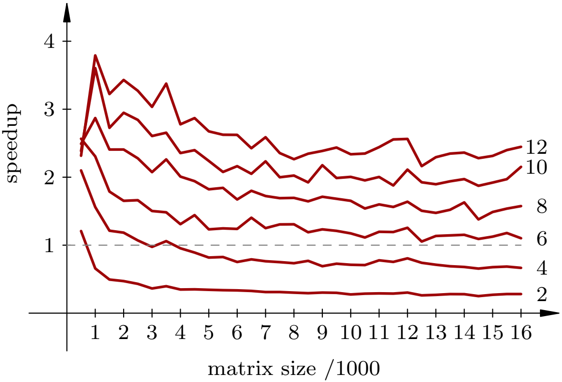
Shared memory algorithms

- Implementation: OpenMP in Fortran routines.

	with sequential MKL	
k	`P-HZ-FB-32`	`P-HZ-BO-32`
500	1.41	0.88
1000	4.78	2.02
1500	14.57	5.99
2000	30.02	12.13
2500	53.13	22.34
3000	86.78	36.08
3500	129.37	55.20
4000	180.32	86.36
4500	249.92	119.74
5000	320.39	159.59

Times (in seconds).

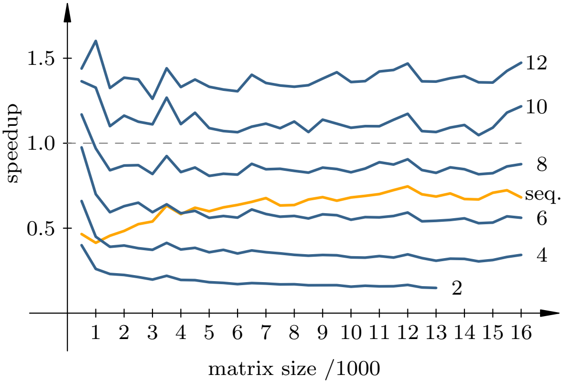
Shared memory algorithms



block-oriented algorithms

Speedup of the shared memory block-oriented algorithms on 2–12 cores vs. the sequential block-oriented Hari–Zimmermann algorithm (threaded MKL on 12 cores).

Shared memory algorithms

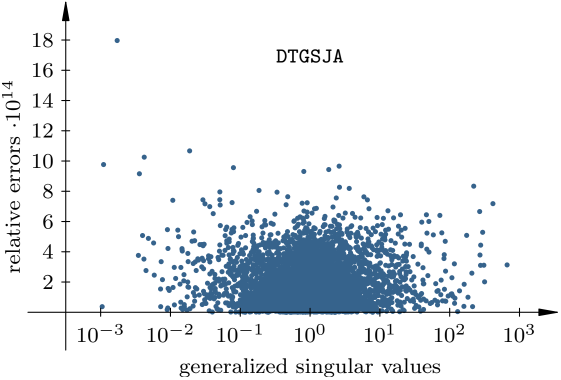


full block algorithms

Speedup of the shared memory full block algorithms on 2–12 cores vs. the sequential block-oriented Hari–Zimmermann algorithm (threaded MKL on 12 cores).

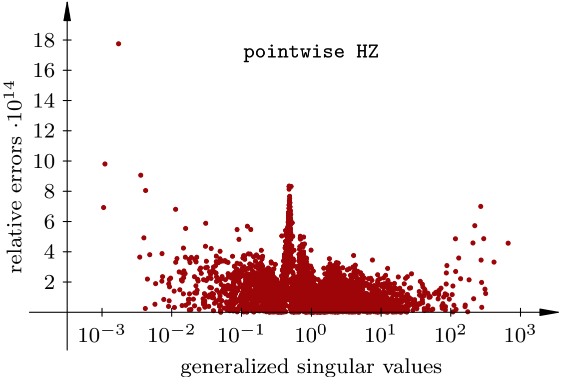
Accuracy (matrix of order 5000)

Test matrix condition number $\max \sigma_i / \min \sigma_i \approx 6.32 \cdot 10^5$



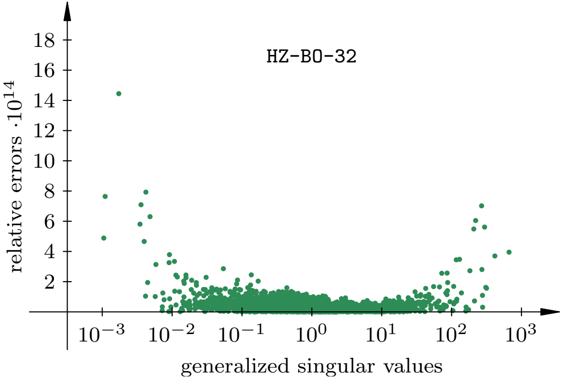
Accuracy (matrix of order 5000)

Test matrix condition number $\max \sigma_i / \min \sigma_i \approx 6.32 \cdot 10^5$



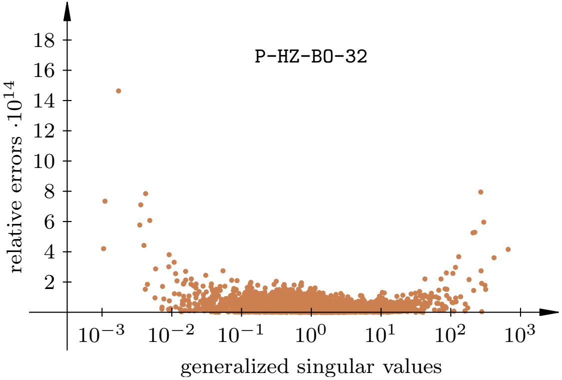
Accuracy (matrix of order 5000)

Test matrix condition number $\max \sigma_i / \min \sigma_i \approx 6.32 \cdot 10^5$



Accuracy (matrix of order 5000)

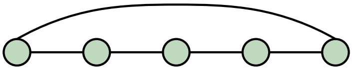
Test matrix condition number $\max \sigma_i / \min \sigma_i \approx 6.32 \cdot 10^5$



Distributed memory algorithms

Distributed algorithms = another level of hierarchy added

- shared-memory algorithm — a building block for the distributed memory algorithm (hybrid MPI/OpenMP)
- only conceptual difference between the distributed memory and the shared memory HZ algorithm — exchange updated block-columns among the MPI processes
- Cartesian topology — one dimensional torus of processes.



- each MPI process in each step sends only one block-column and receives only one block column.

Distributed vs. shared memory algorithms

	number of	time
MPI processes	cores	`MPI-HZ-BO-32`
2	24	15323.72
4	48	8229.32
6	72	6049.77
8	96	4276.65
10	120	3448.90
12	144	3003.39
14	168	2565.29
16	192	2231.71

The running times of the hybrid MPI/OpenMP version HZ, matrix pair of order 16000.

Distributed vs. shared memory algorithms

number of cores	time	
	`P-HZ-FB-32`	`P-HZ-BO-32`
2	–	42906.93
4	35168.73	18096.72
6	21473.00	10936.10
8	13745.17	7651.86
10	9901.96	5599.25
12	8177.90	4925.56

The running times for the full block and block-oriented shared memory algorithms for the same matrix.

Conclusion

On a particular hardware (with threaded MKL on 12 cores)

- Pointwise HZ method is 3 times faster than `DTGSJA` on matrices of order 5000.
- Sequential block-oriented `HZ-BO-32` algorithm is 15 times faster than the pointwise algorithm, i.e., more than 47 times faster than `DTGSJA`.
- For the fastest, explicitly parallel, shared memory algorithm `P-HZ-BO-32`, the speedup factor is 126!
- `DTGSJA` is unable to handle large matrices in any reasonable time.
- Triangularization is mandatory for `DTGSJA`, but not necessary for the Hari–Zimmermann method, when G is of full column rank.