Parallel solution of the generalized eigenvalue problem given in a factored form

Edoardo di Napoli[1], Vedran Novaković[2], Gayatri Čaklović[3], Sanja Singer[4]

[1] Jülich Supercomputing Centre, and RWTH Aachen, Germany

[2] Universitat Jaume I, Castellón de la Plana, Spain

[3] PhD. student at Jülich Supercomputing Centre, Germany

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

Czech Academy of Sciences,
November 9, 2018, Prague, Czech Republic

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

Outline of the talk:

- description of the problem,
- solution of the problem in three steps
  - step 1 — Hermitian indefinite factorizations
  - optional step 2 — hyperbolic QR factorization (JQR)
  - step 3 — implicit HZ algorithm for the hyperbolic generalized SVD
- implementation details of the parallel algorithm,
- partial results of numerical testing.

Problem description

Density Functional Theory framework

- is used in simulation of the physical properties of complex quantum mechanical systems made of few dozens up to few hundreds of atoms
- the core of the method relies on the simultaneous solution of a set of Schrödinger–like equations also known as Kohn–Sham equations
- there exists a wide variety of approaches that can be used to "translate" the DFT mathematical layout into a computational tool.

Full-potential Linearized Augmented Plane Wave (FLAPW) method

- FLAPW method is one of the most accurate methods — particular discretization of the DFT fundamental equations
- FLAPW is all-electron method — it explicitly describes all of the (potentially large number of) electrons in the material with a much larger number of basis function
- it is a quite computationally expensive method.

FLAPW method

Full-potential Linearized Augmented Plane Wave (FLAPW) method

- the discretization in FLAPW method leads to the solution of the generalized eigenvalue problem for matrices (H, S), where

$$
\begin{aligned}
H = \sum_{a=1}^{N_A} (A_a^* T^{[AA]} A_a + A_a^* T^{[AB]} B_a \\
+ B_a^* T^{[BA]} A_a + B_a^* T^{[BB]} B_a)
\end{aligned}
$$

$$
S = \sum_{a=1}^{N_A} (A_a^* A_a + B_a^* U_a^* U_a B_a),
$$

where $A_a, B_a \in \mathbb{C}^{N_L \times N_G}$, $T_a^{[\cdots]} \in \mathbb{C}^{N_L \times N_L}$, $U \in \mathbb{C}^{N_L \times N_L}$ is a diagonal matrix, while $(T^{[AA]})^* = T^{[AA]}$, $(T^{[BB]})^* = T^{[BB]}$, and $(T^{[AB]})^* = T^{[BA]}$.

Problem sizes

Typical matrix sizes

- $N_A = \mathcal{O}(100)$, $N_G = \mathcal{O}(1000)$–$\mathcal{O}(10000)$, and $N_L = \mathcal{O}(100)$
- test examples NaCl – $N_A = 512$, $N_L = 49$
  - $N_G = 2256$, $N_G = 3893$, $N_G = 6217$, $N_G = 9273$
- test examples AuAg – $N_A = 128$, $N_L = 121$
  - $N_G = 3275$, $N_G = 5638$, $N_G = 8970$, $N_G = 13379$.

Computation of H and S

Proposed by Fabregat–Traver at al.

- write H as $H = H_{AA} + H_{AB+BA+BB}$

$$H_{AA} = \sum_{a=1}^{N_A} A_a^* T^{[AA]} A_a$$

$$H_{AB+BA+BB} = \sum_{a=1}^{N_A} \left(B_a^* T^{[BA]} A_a + A_a^* T^{[AB]} B_a + B_a^* T^{[BB]} B_a \right)$$

$$= \sum_{a=1}^{N_A} (B_a^* Z_a + Z_a^* B_a)$$

(`ZHER2K`s!), where

$$Z_a = T^{[BA]} A_a + \frac{1}{2} T^{[BB]} B_a.$$

Transformation of the problem

Modification?

Why

- the algorithm proposed by Fabregat–Traver at al. computes in parallel only H and S — then use any GEVD,
- intention to keep matrices in a factored form – ideal for parallelization of the GEVD
- usage of one-sided methods — faster than the two-sided methods — columnwise action
- such approach usually computes small eigenvalues more accurately
- similar algorithm for the (real) generalized SVD is approximately 125 times faster than the LAPACK routine with threaded MKL.

Transform the problem!

Transformed problem

- by using the properties of matrices $T^{[\cdots]}$ it is obvious that the problem can be written as

$$H = \sum_{a=1}^{N_A} \begin{bmatrix} A_a^* & B_a^* \end{bmatrix} \begin{bmatrix} T^{[AA]} & T^{[AB]} \\ (T^{[AB]})^* & T^{[BB]} \end{bmatrix} \begin{bmatrix} A_a \\ B_a \end{bmatrix} := \sum_{k=1}^{n} H_k^* T_k H_k,$$

$$S = \sum_{a=1}^{N_A} \begin{bmatrix} A_a^* & B_a^* U_a^* \end{bmatrix} \begin{bmatrix} A_a \\ U_a B_a \end{bmatrix} := \sum_{k=1}^{n} S_k^* S_k.$$

Transform the problem!

... or as products of three (two) matrices

$$H = \begin{bmatrix} H_1^* & \cdots & H_n^* \end{bmatrix} \begin{bmatrix} T_1 & & \\ & \ddots & \\ & & T_n \end{bmatrix} \begin{bmatrix} H_1 \\ \vdots \\ H_n \end{bmatrix} := \widetilde{F}^* T \widetilde{F}$$

$$S = \begin{bmatrix} S_1^* & \cdots & S_n^* \end{bmatrix} \begin{bmatrix} S_1 \\ \vdots \\ S_n \end{bmatrix} := G^* G.$$

Matrix sizes

- $H_k, S_k \in \mathbb{C}^{(2N_L)\times N_G}$, $T_k \in \mathbb{C}^{(2N_L)\times(2N_L)}$,
- $F, G \in \mathbb{C}^{(2N_A N_L)\times N_G}$, $T \in \mathbb{C}^{(2N_A N_L)\times(2N_A N_L)}$.

Transform the problem!

Make T simpler

- the method can be applied even on already described matrices $\widetilde{F}$, G and T implicitly, but multiplication by T is slow
- T should be either factored by using somewhat modified Hermitian indefinite factorization (HIF) — simultaneous factorizations of all T_ks, or
- T could be diagonalized (simultaneous diagonalization of T_ks) — diagonalization is too slow
- therefore, H could be written as

$$H := F^* J F, \qquad J = \mathrm{diag}(\pm 1).$$

Step 1

Hermitian indefinite factorization

'Cholesky-like' factorization for indefinite matrices?

- Factorization exists in the following form for any indefinite A

$$A = P^T R^T D R P$$

- 2×2 blocks in D or 2×2 diagonal blocks in R are permitted
- problem: 1×1 or 2×2 block at the pivot position could be singular
- solution: two-sided pivoting (permutation matrix P). If
  - all diagonal elements are 0 and
  - all principal submatrices of order 2 are singular

  then $A = 0$.
- factorization with 2×2 blocks in D is constructed by James Bunch in his PhD thesis (1969)

Modified Hermitian indefinite factorization

From HIF to modified HIF

- modified form, with diagonal $D = \mathrm{diag}(\pm 1)$ is given by Ivan Slapničar in his PhD thesis (1992)
- if d_{ii} is 1×1 block — leave $\mathrm{sign}(d_{ii})$ on the diagonal, and multiply i-th row of R by $\sqrt{|d_{ii}|}$
- if D_{ii} is 2×2 block — diagonalize D_{ii} by a single Jacobi rotation, multiply both rows with this rotation, and repeat previous step for both rows
- now D has 1 or -1 on its diagonal, and, in the case of 2×2 blocks in the original D, R has 2×2 blocks on its diagonal

Examples and problems of HIF

Example

For nonsingular matrix

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \\ 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}$$

- there is no 1×1 pivots (diagonal is 0);
- we need symmetric pivoting since 2×2 block at the pivot position is singular
- Stability reasons – swap 'most nonsingular block' at pivot position — swap first and fourth row and column.

Examples and problems of HIF

Example

For nonsingular matrix

$$A = \begin{bmatrix} 0 & 4 & 0 \\ 4 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- pivoting is not needed, but
- factorization is not unique, for example if $D = \mathrm{diag}(\pm 1)$, then

$$A = \begin{bmatrix} 0 & 4 & 0 \\ 4 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha & \beta & 0 \\ \alpha & -\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}^* \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha & \beta & 0 \\ \alpha & -\beta & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

for all $\alpha \cdot \beta = 2$.

Pivoting in the Hermitian indefinite factorizations

Complete pivoting

- Described in the quite famous paper by Bunch and Parlett (1971)
- define $\mu_1 = \max_i |a_{ii}|$, $\mu_0 = \max_{i,j} |a_{ij}|$, $\alpha = (1 + \sqrt{17})/8$
- if $\mu_1 \geq \alpha \cdot \mu_0$ then choose 1×1 pivot, otherwise 2×2 pivot
- this pivoting is based on the smallest element growth
- Kahan's suggestion (in letter to R. de Meersman and L. Schotsmans, (1965)) — take pivot block as the 'most nonsingular' block, i.e., with biggest μ_c

$$\mu_c = \max_{i,j,k} \{|a_{ii}|, |a_{jj} a_{kk} - |a_{jk}|^2|\}$$

- $\mathcal{O}(n^4)$ operations needed to find pivot if A is given implicitly, by its factors

Pivoting in the Hermitian indefinite factorizations

Partial pivoting

- Described in the papers by Bunch, Kaufmann and Parlett (1976–1977)
- little bit more complicated to describe
- searches 1×1 pivots on the whole diagonal
- in the case of 2×2 pivots, searches only for the new second row (first row is fixed)
- advantage $\mathcal{O}(n^3)$ operations needed to find pivot if A is given implicitly, by its factors
- disadvantage — less stable than complete pivoting

Step 2 (optional)

Optional step

Optional step — make J, F and G square

- square matrix – faster third step — HZ algorithm
- this step is the hyperbolic QR factorization on F and the QR factorization on G – both algorithms are moderately parallel
- pivoting strategy – partial pivoting?, threshold pivoting?
- usage of (block)-reflectors, Hyperbolic URV (HURV) factorization, or (block)-rotations?
- do it or not – depends on the ratio (number of rows) / (number of columns) of F and G

Hyperbolic QR factorization

Definition

Let $F \in \mathbb{C}^{m \times n}$, $m \geq n$, and $J \in \mathbb{C}^{m \times m}$, $J = \mathrm{diag}(j_{11}, \ldots, j_{mm})$, $j_{ii} \in \{1, -1\}$, be given matrices, such that $A := F^* J F$ is nonsingular. Factorization

$$F = P_1 Q R P_2^* = P_1 Q \begin{bmatrix} R_1 \\ 0 \end{bmatrix} P_2^*, \quad Q^* J' Q = J', \quad J' = P_1^* J P_1,$$

where P_1 and P_2 are permutation matrices, matrix Q is J'-unitary, and R_1 is block upper triangular with diagonal blocks of order 1 or 2, is called the hyperbolic QR factorization of F according to J.

Reflectors

Definition

A matrix $H \in \mathbb{C}^{m\times m}$ will be called a J-reflector if it is J-unitary, J-Hermitian and a reflector, i.e.,

$$H^* J H = J, \quad JH = H^* J, \quad H^2 = I.$$

Proposition

Let $J \in \mathbb{C}^{m\times m}$ be a given nonsingular matrix. Any two equalities in the definition imply the third one.

Nondegenerate matrix

Matrix $W \in \mathbb{C}^{m \times p}$ is nondegenerate (with respect to J) if and only if range of W, $\mathcal{R}(W)$ is nondegenerate.

Proposition

The following statements are equivalent:

- W is nondegenerate,
- $\mathcal{N}(W^* J W) = \mathcal{N}(W)$,
- $\mathcal{R}(W^* J^* W) = \mathcal{R}(W^*)$,
- $\operatorname{rank}(W^* J W) = \operatorname{rank}(W)$.

Simple and block reflectors

Block J-reflector

For any given $W \in \mathbb{C}^{m \times p}$, a matrix $H \in \mathbb{C}^{m \times m}$ defined by

$$H = H(W) = I - 2W(W^*JW)^+W^*J$$

will be called a block J-reflector (generated by W).

If $p = 1$ (W is a vector), H is sometimes called a simple J-reflector (Bunse Gerstner).

Proposition

$H = H(W)$ satisfies all three reflector properties, i.e., it is a J-reflector.

Reflection properties

Theorem
Let $H = H(W)$ be a block J-reflector generated by W.
If W is nondegenerate, then

$$\begin{aligned} Hx &= -x, \quad \text{for all } x \in \mathcal{R}(W), \\ Hy &= y, \quad \text{for all } y \in \mathcal{R}(W)^{[\perp]}, \end{aligned}$$

i.e., H reflects (reverses) $\mathcal{R}(W)$ with respect to $\mathcal{R}(W)^{[\perp]}$.

If W is degenerate then

$$Z = Z(W) = W(W^*JW)^+$$

is nondegenerate and $H(Z) = H(W)$.
Moreover, $\mathcal{R}(Z) \subseteq \mathcal{R}(W)$, and $\mathcal{R}(Z) = \mathcal{R}(W)$ holds if and only if W is nondegenerate.

A few comments

- If $p = 1$ it is easy to see that $w \neq 0$ is nondegenerate if and only if w is nonisotropic, i.e., $w^* J w \neq 0$
- For $p \geq 2$, nondegeneracy becomes less restrictive than nonisotropy
- $\mathcal{R}(W)$ may contain some nonzero isotropic vectors
- As long as none of these vectors are J-perpendicular to the whole subspace, W is nondegenerate and $H(W)$ reflects the whole range of W.
- This fact will be used later (with $p = 2$) to obtain the hyperbolic QR factorization.

The mapping problem

For given matrices $F, R \in \mathbb{C}^{m\times q}$, $q \geq 1$, we seek a block J-reflector $H = H(W)$ such that

$$HF = R.$$

If H exists, F and R satisfy

(i) J-isometry property:

$$F^*JF = R^*JR, \tag{1}$$

(ii) J-symmetry property (symmetry with respect to J):

$$R^*JF = F^*JR. \tag{2}$$

Obviously, if R satisfies these conditions, so does $-R$.

An example

Note, if $J = I$ requirements (1) and (2) are necessary and sufficient (Schreiber–Parlett). If $J \neq I$:

- (1) and (2) are necessary for the existence of H, but may not be sufficient.

Example

Let $J = \text{diag}(1, -1, 1, -1)$ be the hyperbolic scalar product and

$$F = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Obviously, $\text{rank}(F) = 2$, $\text{rank}(R) = 1$, and it is easy to verify that both (1) and (2) are satisfied by F and R.

Sum S and difference D

Let D (for "difference") and S (for "sum") be the matrices defined by

$$D = R - F, \quad S = R + F.$$

Lemma

If $F, R \in \mathbb{C}^{m \times q}$ satisfy (1) and (2), then D and S satisfy

$$D^* JS = 0.$$

Mapping theorem 1

J-reflector partial mapping theorem

Let F and R be two matrices in $\mathbb{C}^{m \times q}$ which satisfy (1) and (2). Then $H(D)F = R$ if and only if D is nondegenerate, i.e., D satisfies the rank condition

$$\text{rank}(D^* JD) = \text{rank}(D).$$

Furthermore, $H(S)F = -R$ if and only if S is nondegenerate, i.e., S satisfies the rank condition

$$\text{rank}(S^* JS) = \text{rank}(S).$$

When $J \neq I$, this Theorem gives only sufficient conditions for the existence of a mapping reflector H.

An example

Example

Let $J = \mathrm{diag}(1, -1, 1, -1)$ and

$$F = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

$\mathrm{rank}(F) = \mathrm{rank}(R) = 1$, (1)–(2) are satisfied, but $D^* J D = S^* J S = 0$, and we cannot construct H by using either D or S. On the other hand, $H(W)F = R$ with

$$W = \begin{bmatrix} 0 & 1/4 \\ 0 & -3/4 \\ -1 & -2 \\ -1 & 0 \end{bmatrix}.$$

Mapping theorem 2

J-reflector full mapping theorem

Let F and R be two matrices in $\mathbb{C}^{m\times q}$ which satisfy (1) and (2). There exists a J-reflector H such that

$$H(W)F = R,$$

if and only if

$$\mathcal{R}(D) \cap \mathcal{R}(S) = \{0\}.$$

Single column reduction

- If a single pivot column f_1 is chosen then

$$e_k^* J e_k = \mathrm{sign}(f_1^* J f_1) \neq 0.$$

- In the case of two pivot columns

$$F = [f_1, f_2] = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix},$$

where F_1 is 2×2 matrix,

$$M := F^* J F$$

is nonsingular, indefinite, with nonzero offdiagonal elements.

Reduction of two columns

We seek a hyperbolic block reflector H such that $HF = R$. R should be determined such that:

- R and J have the special structures,

$$R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}, \quad J = \mathrm{diag}(J_1, J_2),$$

  R_1 is 2×2 matrix, $J_1 = \mathrm{diag}(j_{11}, -j_{11})$,

- R and F satisfy (1)–(2).

We hope that this system of equations can be solved for R_1, and H follows from the partial mapping theorem.

Reduction of two columns

$M = F^*JF$ is indefinite $\Longrightarrow$ we can find a row permutation P_1 such that

- F_1 is nonsingular and
- $J_1 = \mathrm{diag}(j_{11}, -j_{11})$.

Now

$$HF = H \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$$

implies

$$R_1^* J_1 R_1 = F^* JF = M, \qquad R_1^* J_1 F_1 = F_1^* J_1 R_1.$$

Note that R_1 is nonsingular, so we define

$$K := F_1 R_1^{-1}.$$

Reduction of two columns

In terms of K we have

$$KJ_1K^* = F_1M^{-1}F_1^*, \qquad J_1K = K^*J_1,$$

so K is J-Hermitian, and the first equation can be written as

$$K^2 = F_1M^{-1}F_1^*J_1.$$

Since all matrices are nonsingular square root K always exists, but it not need to be primary square root.

It is difficult to show that either D or S is nondegenerate. Possible solution: swap rows and/or columns of F to make K primary square root.

Alternative reduction of two columns

Since M is Hermitian and indefinite

- it can be diagonalized by a singe Jacobi rotation, U i.e., columns of J are orthogonalized,
- therefore, after the application of this rotation

$$\widehat{F} = FU$$

- we can proceed with two simple J-reflectors that annihilates f_1, and f_2
- these Jacobi rotations U^* are either stored, so we obtain HURV factorization, or can be applied back to the columns r_1 and r_2.

Step 3

One-sided vs. two-sided method

- the original Hari–Zimmerman method works from both sides on the Hermitian matrix pair
- the modified method works from one side on the factors of the Hermitian matrix pair
- idea: think two-sided, act one-sided
- transformations will be computed from the pivot submatrices H_{pq} of H and S_{pq} of S

$$H_{pq} = \begin{bmatrix} F_p^* J F_p & F_p^* J F_q \\ & F_q^* J F_q \end{bmatrix}, \quad S_{pq} = \begin{bmatrix} G_p^* G_p & G_p^* G_q \\ & G_q^* G_q \end{bmatrix}.$$

The complex Hari–Zimmermann method for the GEP

The method consists of 3 transformations (Hari)

- as a preprocessing step H and S can be scaled by the diagonal matrix D such that $\operatorname{diag}(DSD) = I$

$$H_0 := DHD, \quad S_0 := DSD,$$
$$D = \operatorname{diag}\left(\frac{1}{\sqrt{s_{11}}}, \frac{1}{\sqrt{s_{22}}}, \ldots, \frac{1}{\sqrt{s_{nn}}}\right)$$

- in the first step the pivot submatrix $\widehat{S}_0$ of S_0 is diagonalized by the complex rotation

$$\widehat{R}_1 = \begin{bmatrix} \cos\varphi_1 & e^{i\beta_1}\sin\varphi_1 \\ -e^{-i\beta_1}\sin\varphi_1 & \cos\varphi_1 \end{bmatrix},$$

The complex Hari–Zimmermann method for the GEP

The transformations

- the first transformation is

$$H_1 = R_1^* H_0 R_1, \quad S_1 = R_1^* S_0 R_1,$$

  $R_1 = I$ except at the pivot positions, where $R_1 = \widehat{R}_1$.
- if H and S are preprocessed, then $\varphi_1 = -\frac{\pi}{4}$
- in the second step – the diagonal of S_1 is rescaled to I
- this transformation is similar to the preprocessing step

$$H_2 := D_2 H_1 D_2, \quad S_2 := D_2 S_1 D_2.$$

The transformations

- in the third step the pivot submatrix $\widehat{H}_2$ of H_2 is diagonalized by the complex rotation

$$\widehat{R}_3 = \begin{bmatrix} \cos\varphi_3 & e^{i\alpha_3}\sin\varphi_3 \\ -e^{-i\alpha_3}\sin\varphi_3 & \cos\varphi_3 \end{bmatrix},$$

- the third transformation is

$$H_3 = R_3^* H_2 R_3, \quad S_3 = R_3^* S_2 R_3,$$

  $R_3 = I$ except at the pivot positions, where $R_3 = \widehat{R}_3$.
- if H and S are preprocessed, then $\varphi_3 = \vartheta + \frac{\pi}{4}$.

The transformations

- note that after the first three steps, the pivot submatrix $\widehat{S}_3$ is still diagonal (in fact identity)

$$\widehat{S}_3 = \widehat{Z}^* \widehat{S} \widehat{Z}, \quad \widehat{Z} = \widehat{R}_1 \widehat{D}_2 \widehat{R}_3$$

- if H and S are preprocessed, the fourth step is only formal — it helps in coupling together all the transformations

$$H_4 = \Phi_4^* H_3 \Phi_4, \quad S_4 = \Phi_4^* S_3 \Phi_4, \quad \widehat{\Phi}_4 = \mathrm{diag}(e^{-i\sigma_p}, e^{-i\sigma_q}).$$

HZ transformations

$$
\begin{array}{|c|c|}
\hline
* & * \\
\hline
* & * \\
\hline
\end{array}
\qquad\qquad
\begin{array}{|c|c|}
\hline
1 & * \\
\hline
* & 1 \\
\hline
\end{array}
$$

HZ transformations

$$\begin{bmatrix} * & * \\ * & * \end{bmatrix} \qquad \begin{bmatrix} * & 0 \\ 0 & * \end{bmatrix}$$

HZ transformations

$*$	$*$
$*$	$*$

1	0
0	1

HZ transformations

$*$	0
0	$*$

1	0
0	1

The complex Hari–Zimmermann method for the GEP

The coupled transformation Z ...

- looks similar to an ordinary plane rotation: it is the identity matrix, except for its (p,q)-restriction $\widehat{Z}$, where

$$\widehat{Z} = \frac{1}{\sqrt{1-\left(|s_{pq}|\right)^2}} \begin{bmatrix} \cos\varphi & e^{i\alpha}\sin\varphi \\ -e^{-i\beta}\sin\psi & \cos\psi \end{bmatrix},$$

- φ and ψ are determined so that the transformations diagonalize the pivot submatrices $\widehat{H}$ and $\widehat{S}$
- the transformation keeps the diagonal elements of S intact
- if $S = I$ then Z 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 $\widehat{Z}$

- let

$$s = |s_{pq}|, \quad t = \sqrt{1 - s^2}, \quad r = s_{qq} - s_{pp},$$

$$\sigma = \begin{cases} 1 & e \geq 0 \\ -1 & e < 0, \end{cases}, \quad u + iv = e^{-i\arg(s_{pq})} h_{pq},$$

- then if $\gamma = \alpha - \beta$

$$\tan(\gamma) = 2\frac{v}{r}, \qquad -\frac{\pi}{2} \leq \gamma \leq \frac{\pi}{2}$$

$$\tan(2\vartheta) = \sigma \frac{2u - (h_{pp} + h_{qq})s}{\sqrt{e^2 + 4v^2} \cdot t}, \qquad -\frac{\pi}{4} < \vartheta \leq \frac{\pi}{4}$$

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

- and

$$2\cos^2\varphi = 1 + s\sin(2\vartheta) + t\cos(2\vartheta)\cos(\gamma), \quad 0 \le \varphi < \frac{\pi}{2}$$

$$2\cos^2\psi = 1 - s\sin(2\vartheta) + t\cos(2\vartheta)\cos(\gamma), \quad 0 \le \psi < \frac{\pi}{2}$$

$$e^{i\alpha}\sin(\varphi) = \frac{(\sin(2\vartheta) - s) + i\sqrt{1-s^2}\sin(\gamma)\cos(2\vartheta)}{1 - s\sin(2\vartheta) + \sqrt{1-s^2}\cos(\gamma)\cos(2\vartheta)}$$

$$e^{-i\beta}\sin(\psi) = \frac{(\sin(2\vartheta) + s) - i\sqrt{1-s^2}\sin(\gamma)\cos(2\vartheta)}{1 + s\sin(2\vartheta) + \sqrt{1-s^2}\cos(\gamma)\cos(2\vartheta)}.$$

The implicit HZ algorithm

$Z = I$; $\qquad it = 0$

`repeat` // sweep loop

$\quad it = it + 1$

$\quad$ `for` all pairs (p, q), $1 \le p < q \le k$

$\qquad$ compute

$$\widehat{H} = \begin{bmatrix} f_p^* J f_p & f_p^* J f_q \\ & f_q^* J f_q \end{bmatrix}; \qquad \widehat{S} = \begin{bmatrix} g_p^* g_p & g_p^* g_q \\ & g_q^* g_q \end{bmatrix}$$

$\qquad$ compute the elements of $\widehat{Z}$

$\qquad\quad$ // transform F, G and Z

$\qquad [f_p, f_q] = [f_p, f_q] \cdot \widehat{Z}$

$\qquad [g_p, g_q] = [g_p, g_q] \cdot \widehat{Z}$

$\qquad [z_p, z_q] = [z_p, z_q] \cdot \widehat{Z}$

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

Parallelization and numerical testing

Hardware platform

Developer Edition of the Intel Xeon Phi 7210 (KNL) processor

- 96 GB of RAM per node,
- 64 cores per node,
- clock 1.30 GHz (Turbo Boost off),
- Intel AVX-512 (Advanced Vector Extensions) instruction set
- presence of two vector processing units (VPUs) per core — each VPU operates independently on 512-bit vector registers – suitable for simultaneous processing of 16 `single precision` or 8 `double precision` numbers.

The first step

Hermitian indefinite factorization of all T_k's

- `for all` T_k`'s do in parallel`

$$T_k = P_k^T R_k^* D_k R_k P_k,$$

P_k is a permutation matrix – formed as in LAPACK
(as a sequence of partial permutations),
D_k is block diagonal, with 1×1 or 2×2 diagonal blocks,
R_k is upper triangular

- pivoting — complete pivoting used (numerical stability)

- `diagonalize all` D_k`'s in parallel`

$$D_k = U_k^* \Delta_k U_k = U_k^* \sqrt{|\Delta_k|} J_k \sqrt{|\Delta_k|} U_k,$$

Δ_k diagonal, U_k block-diagonal, unitary, $J_k = \mathrm{diag}(\pm 1)$,

The first step (cnt'd)

- `for all` J_k `repermute them in parallel`

$$J_k := \widetilde{P}_k^T \operatorname{diag}(I, -I)\widetilde{P}_k$$

  $\widetilde{P}_k$ is a permutation,

- `multiply rows of all` R_k `and repermute them`

$$R_k = \widetilde{P}_k \sqrt{|\Delta_k|} U_k$$

- `repermute columns of all` R_k `according to permutations stored in` P_k

$$R_k := R_k P_k.$$

The first step (cnt'd)

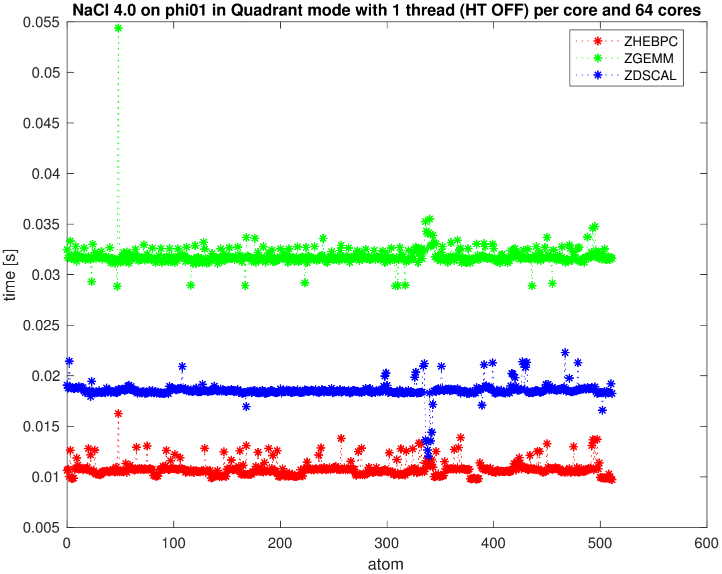
Final state

$$T_k = R_k^* \operatorname{diag}(I, -I) R_k, \quad k = 1, \ldots, n.$$

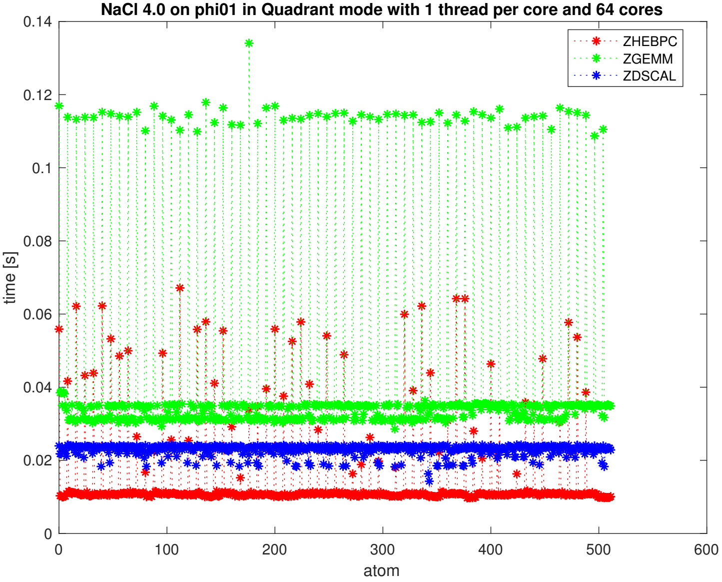
Comments

- in the first step, the factorization is sequential for each T_k
- each physical core of the Xeon Phi deals with one or more T_k in turn (OpenMP `parallel do` over all T_k)
- each core can use its own 1–4 hyperthreads in a call of the threaded BLAS routines – therefore even per core algorithm is somewhat parallel – but do not use hyperthreading, since...

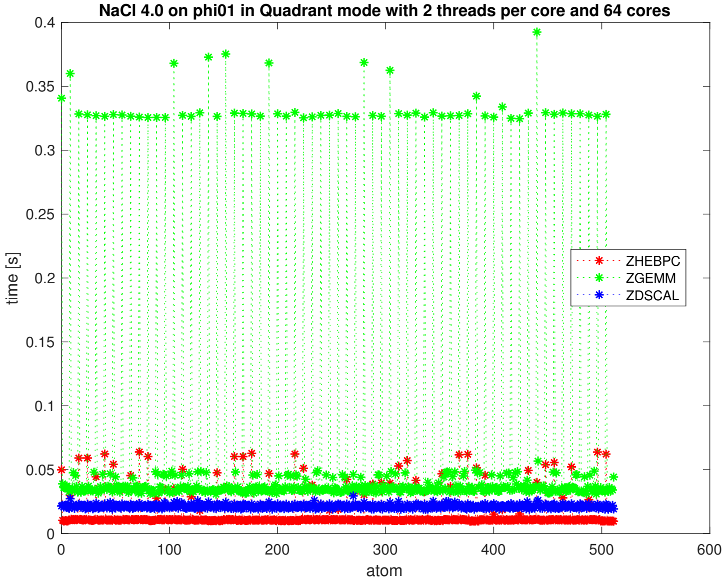
1 thread, HT off, NaCl, $N_L = 49$, $N_A = 512$, $N_G = 9273$



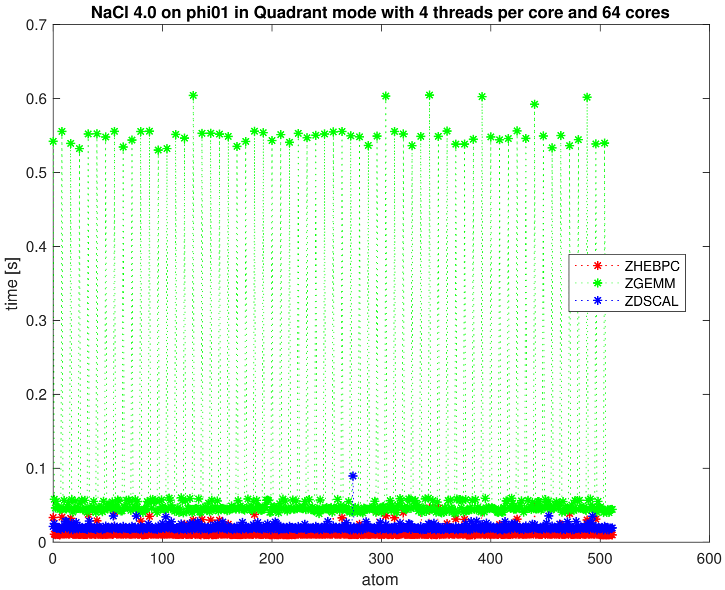
1 thread, NaCl, $N_L = 49$, $N_A = 512$, $N_G = 9273$



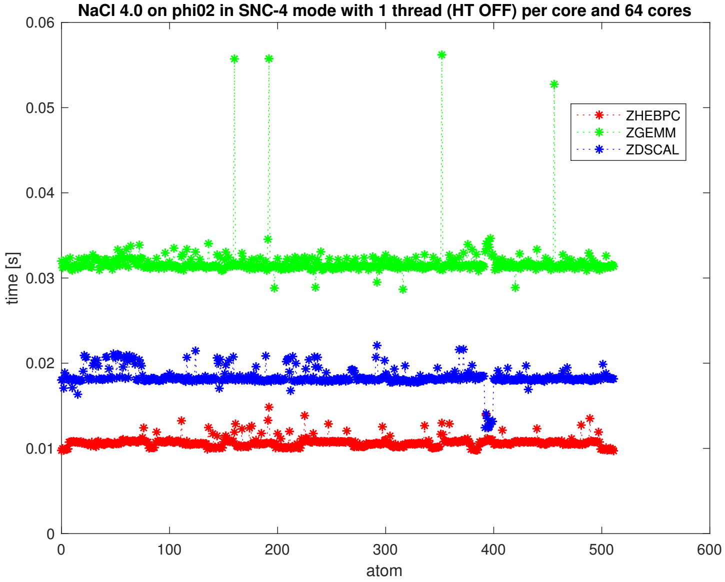
2 threads, NaCl, $N_L = 49$, $N_A = 512$, $N_G = 9273$



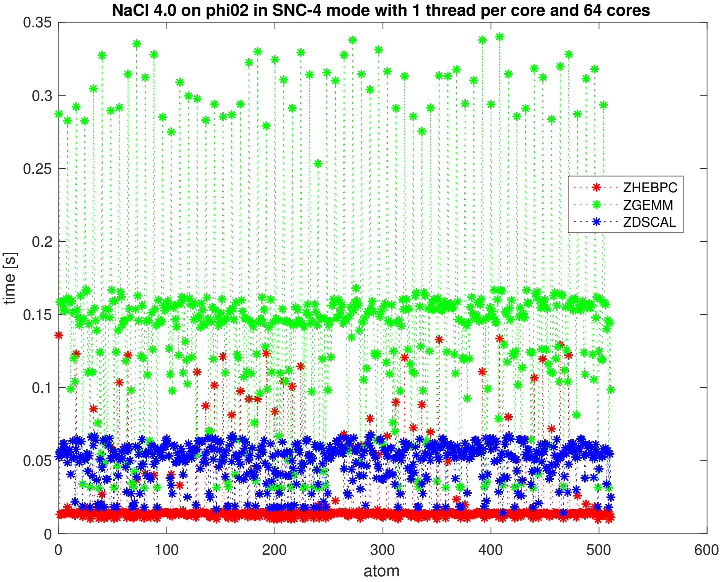
4 threads, NaCl, $N_L = 49$, $N_A = 512$, $N_G = 9273$



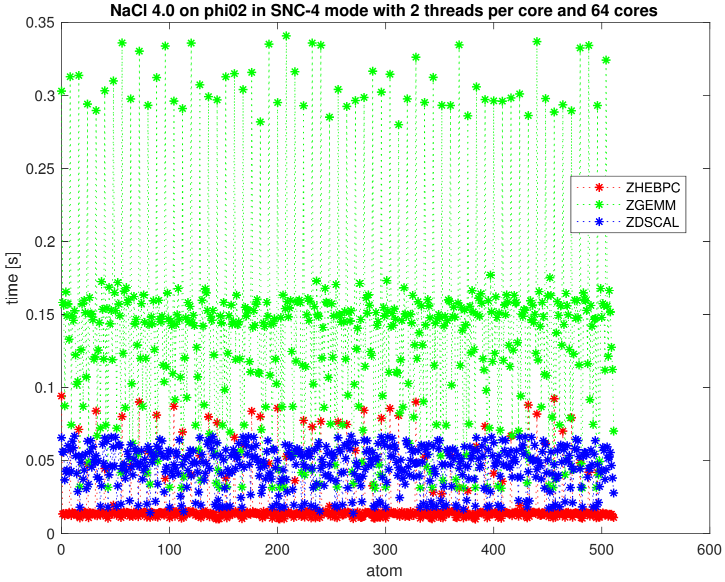
1 thread, HT off, NaCl, $N_L = 49$, $N_A = 512$, $N_G = 9273$



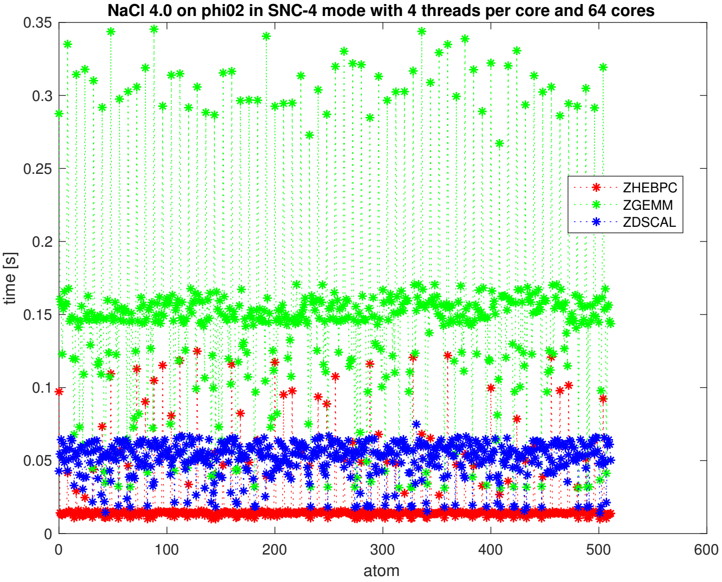
1 thread, NaCl, $N_L = 49$, $N_A = 512$, $N_G = 9273$



2 threads, NaCl, $N_L = 49$, $N_A = 512$, $N_G = 9273$



4 threads, NaCl, $N_L = 49$, $N_A = 512$, $N_G = 9273$



The second and the optional step

Form J, F and G

- `store` $J = \mathrm{diag}(J_1, \ldots, J_n)$,
- `multiply` $F = \mathrm{diag}(R_1, \ldots, R_n)F$, `each` R_k `in parallel`
- `scale` B_k `by` U_k `in parallel and store` G

Optional step — make J, F and G square

- what is the efficient way to compute the JQR?
- note that the second matrix G should shortened by the ordinary QR factorization with the same pivoting as in the JQR
- this factorization should be done by prepermutation (first), and the by the tall-and-skinny QR
- we hope that there is a crossing when JQR + QR + HZ algorithm is faster than the HZ on the rectangular matrices.

Driver level of the implicit HZ algorithm

Details of the level-2 algorithm

- algorithm is Generalized Hyperbolic SVD of (F, G) with respect to J
- matrices F and G are divided in even number of block-columns

$$F = [F_1, \ldots, F_{2b}], \quad G = [G_1, \ldots, G_{2b}]$$

- number of block-columns depend on the number of physical cores of the processor (our case: 64 cores = maximum 128 blocks, no hyperthreading)
- each thread is connected to one physical core.

Driver level of the implicit HZ algorithm

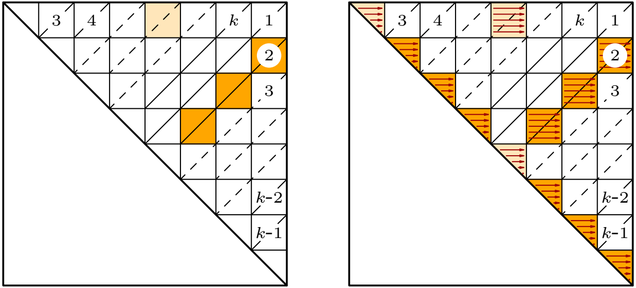
Each thread . . .

- works on a pair of block-columns of each matrix given by some parallel pivot strategy
- allocates storage for $[F_p, F_q]$, $[G_p, G_q]$, their "shadow" counterparts, and for the part of the transformation matrix
- "shadow" memory — used for scaling by J_k and data exchange
- since architecture is NUMA (Non Uniform Memory Access), columns are also physically copied to "shadow" memory (alternative: reassignment of pointers)
- allocates square space in fast MCDRAM for computation of the transformation Z_{pq} and the pivot block submatrices H_{pq} and S_{pq}.

Pivoting strategy

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.

Driver level of the implicit HZ algorithm

Each thread . . .

- actually computes H_{pq} and S_{pq} (ZGEMM and ZHERK)
- factorizes H_{pq} and S_{pq} by the Hermitian indefinite factorization (test of definiteness of S_{pq})

  $$H_{pq} = F_{pq}^* J_{pq} F_{pq}, \quad S_{pq} = G_{pq}^* G_{pq},$$

  where F_{pq}, G_{pq}, and J_{pq} are square
- calls level-1 (non-blocked routine) on the triplet (F_{pq}, G_{pq}, J_{pq})
- applies transformation matrix to original F_{pq}, G_{pq}, and Z_{pq} (ZGEMMs)
- transfers one triplet of (F_ℓ, G_ℓ, Z_ℓ), $\ell \in \{p, q\}$ to the next "owner" (thread) into its "shadow" memory.

Computational level of the implicit HZ algorithm

Details of the level-1 algorithm

- single-threaded (including BLAS calls) SIMD-parallel code,
- the main loop — `sweep` iterations (1, m, until convergence)
- parallel pivot strategy determines maximal number of independent pivot pairs — stage of the algorithm
- in each stage — pairs are divided into groups of 8 pairs (AVX-512 instructions)
- compute 6 dot products (vectorized + reductions) with only 4 accesses of f_p, f_q, g_p, and g_q:

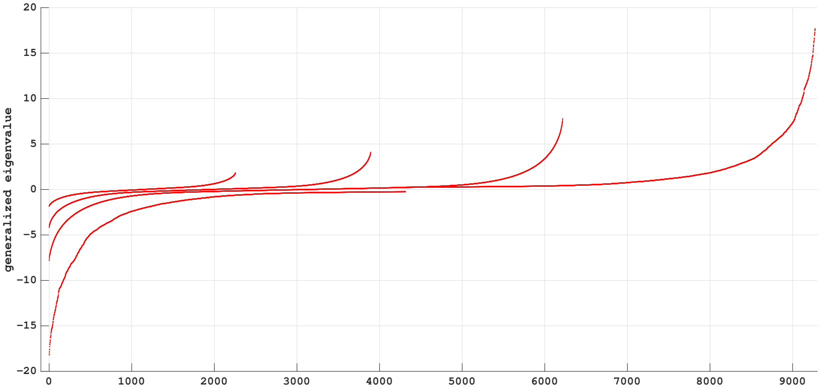
$$\widehat{H}_{pq} = \begin{bmatrix} f_p^* J_p f_p & f_p^* J_q f_q \\ & f_q^* J_q f_q \end{bmatrix}, \qquad \widehat{S}_{pq} = \begin{bmatrix} g_p^* g_p & g_p^* g_q \\ & g_q^* g_q \end{bmatrix},$$

Computational level of the implicit HZ algorithm

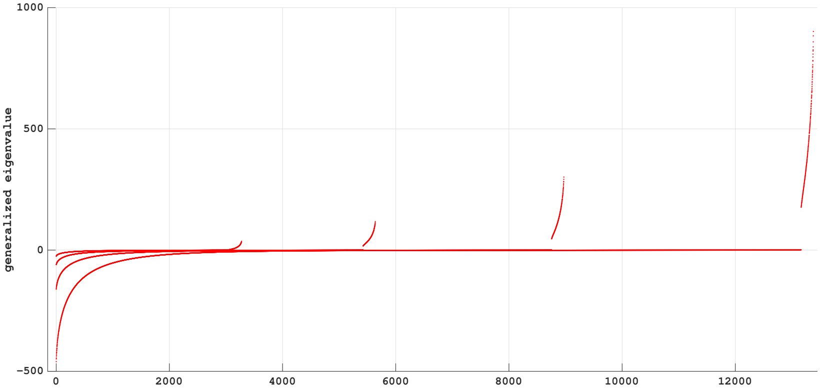
Details of the level-1 algorithm

- an example: the dot products are computed without BLAS to avoid function calls (slow!)
- computing transformation matrices for 8 pairs simultaneously
- transformations to 8 column pairs (f_p, f_q), (g_p, g_q), (z_p, z_q) are applied sequentially for each pair (cache!)

Distribution of eigenvalues – NaCl



Distribution of eigenvalues – AuAg



Timings

Example	Problem size	Number of cores 64	32
NaCl 2.5	50176 × 2256	800.70	556.95
NaCl 3.0	50176 × 3893	1973.64	1465.68
NaCl 3.5	50176 × 6217	2810.50	3660.44
NaCl 4.0	50176 × 9273	4846.98	7028.50
AuAg 2.5	26136 × 3275	724.20	587.23
AuAg 3.0	26136 × 5638	1549.92	1715.60
AuAg 3.5	26136 × 8970	3152.78	4711.65
AuAg 4.0	26136 × 13379	6544.16	11955.74

Conclusion

On a particular hardware testing space is enormous

- use Quadrant or SNC-4 clustering mode?
- in a single step — transform columns only once (block-oriented algorithm) or fully diagonalize them (full block algorithm)
- best pivoting strategy?
- is there need to shorten the columns by the hyperbolic QR factorization, and is there a switching point (use them or not)

  ...

Work in progress

- only lower 20% of the eigenvalues are needed
- is there any sufficiently parallel algorithm to compute them (without multiplication of the factors)?