THE SUPPLEMENTARY MATERIAL FOR THE PAPER "THE LAPW METHOD WITH EIGENDECOMPOSITION BASED ON THE HARI–ZIMMERMANN GENERALIZED HYPERBOLIC SVD"*

SANJA SINGER†, EDOARDO DI NAPOLI‡, VEDRAN NOVAKOVIĆ§, AND GAYATRI ČAKLOVIĆ¶

S. Supplementary material. This is the supplementary material for the sections 4, 6 and 7 of the main paper, to which the reader is encouraged to refer.

S.1. Addendum to section 4. Figure S.1 shows `ZVSCAL`, a Fortran routine similar to the one used in our code, that is a fully vectorized, cache-friendly alternative to applying `ZDSCAL` with a non-unit stride in Algorithm 4.1 in the manuscript.

```
PURE SUBROUTINE ZVSCAL(M, DX, ZY)
  IMPLICIT NONE
  ! alignment in bytes .EQ. cache line size
  INTEGER, PARAMETER :: ALIGNMENT_IN_B = 64
  INTEGER, INTENT(IN) :: M
  DOUBLE PRECISION, INTENT(IN) :: DX(M)
  DOUBLE COMPLEX, INTENT(INOUT) :: ZY(M)

  INTEGER :: I
  !DIR$ ASSUME_ALIGNED DX:ALIGNMENT_IN_B

  !DIR$ VECTOR ALWAYS ASSERT
  DO I = 1, M
     ZY(I) = DX(I) * ZY(I)
  END DO
END SUBROUTINE ZVSCAL
```

FIG. S.1. *`ZVSCAL`, a Fortran routine for scaling the elements of a complex vector `ZY` by the corresponding elements of a real vector `DX`, as a generalization of the `ZDSCAL` BLAS 1 routine.*

Figure S.1 also illustrates the methods of vectorizing loops and specifying the alignment requriements with the Intel Fortran compiler directives, used in our code.

In Algorithm 4.1, instead of N_L `ZDSCAL`s over rows of length N_G, there could be N_G applications of `ZVSCAL`s on the contiguous column sections of length N_L. Timing results of such a variant of Phase 1, obtained on [1], are given in Table S.1. The

*This work has been supported in part by Croatian Science Foundation under the project IP–2014–09–3670, and also in part by a bilateral research project "Optimization of material science algorithms on hybrid HPC platforms" funded by the Croatian Ministry of Science and Education (MZO) and the German Academic Exchange Service (DAAD).

†University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture, I. Lučića 5, 10000 Zagreb, Croatia, (ssinger@fsb.hr).

‡Forschungszentrum Jülich, Jülich Supercomputing Centre, Wilhelm–Johnen-Straße, Jülich, 52425, Germany and RWTH Aachen University, AICES, Schinkelstraße 2, Aachen, 52062, Germany, (e.di.napoli@fz-juelich.de, dinapoli@aices.rwth-aachen.de).

§Completed a majority of his part of the research while being affiliated to Universidad Jaime I, Av. Vicent Sos Baynat, 12071 Castellón de la Plana, Spain, (novakoni@uji.es).

¶Ph.D. student, Forschungszentrum Jülich, Jülich Supercomputing Centre, Wilhelm–Johnen-Straße, Jülich, 52425, Germany, (g.caklovic@fz-juelich.de).

percentage of time spent in ZVSCALs is, on average (but not in each case), lower than in ZDSCALs, indicating the former as a viable choice for Phase 1.

TABLE S.1
The average per-atom wall execution time (wtime) of Phase 1 with 24 and 48 threads. Since the routine weights are rounded to the nearest per mil, their sum may not yield 100%. The first weight corresponds to ZHEBPJ*, the second one to* ZGEMM*, and the third one to* ZVSCALs *step of Phase 1.*

ID	average wtime [s] per atom		routine weights %:%:%	
	24 threads	*48* threads	*24* threads	*48* threads
A1	0.071674	0.112488	64.6 : 33.8 : 1.6	68.4 : 30.1 : 1.5
A2	0.046098	0.049334	15.3 : 80.9 : 3.8	15.5 : 80.9 : 3.5
A3	0.068751	0.072903	10.4 : 85.9 : 3.8	10.2 : 85.8 : 4.0
A4	0.094880	0.097327	7.0 : 89.3 : 3.7	6.9 : 89.5 : 3.5
B1	0.003533	0.004033	15.6 : 78.4 : 6.0	14.4 : 78.9 : 6.7
B2	0.005745	0.006444	9.7 : 83.1 : 7.3	9.0 : 82.4 : 8.6
B3	0.009061	0.010523	6.5 : 85.3 : 8.2	6.6 : 83.7 : 9.7
B4	0.013365	0.015121	4.1 : 87.1 : 8.7	4.8 : 84.2 : 11.1

S.2. Addendum to section 6. In Figures S.2 and S.3 the generalized eigenvalues Λ, obtained by the Phase 3 with 64 threads, are plotted for the entire dataset.

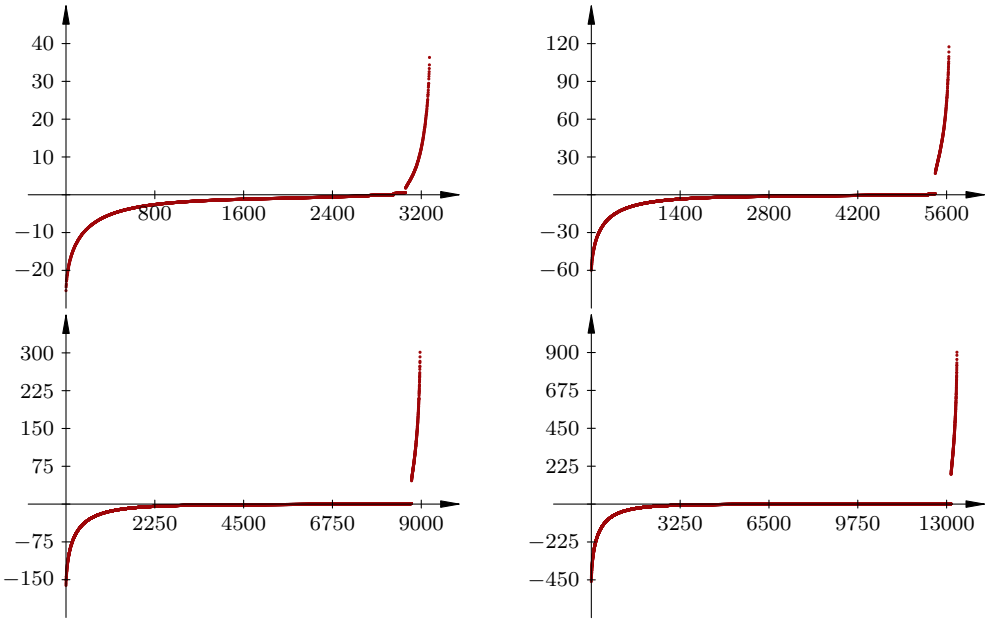

FIG. S.2. *Left to right, top to bottom: the generalized eigenvalues* $\Lambda(A1)$, $\Lambda(A2)$, $\Lambda(A3)$, $\Lambda(A4)$.

Figures S.4 and S.5 show the relative errors $(\widehat{\Lambda}_i - \widetilde{\Lambda}_i)/\widetilde{\Lambda}_i$ in the computed and ascendingly sorted generalized eigenvalues, where $\widetilde{\Lambda}_i$ come from Phase 2 and the Level 2 (BO) Phase 3 algorithm, while $\widehat{\Lambda}_i$ come from the ZHEGV routine in the left,

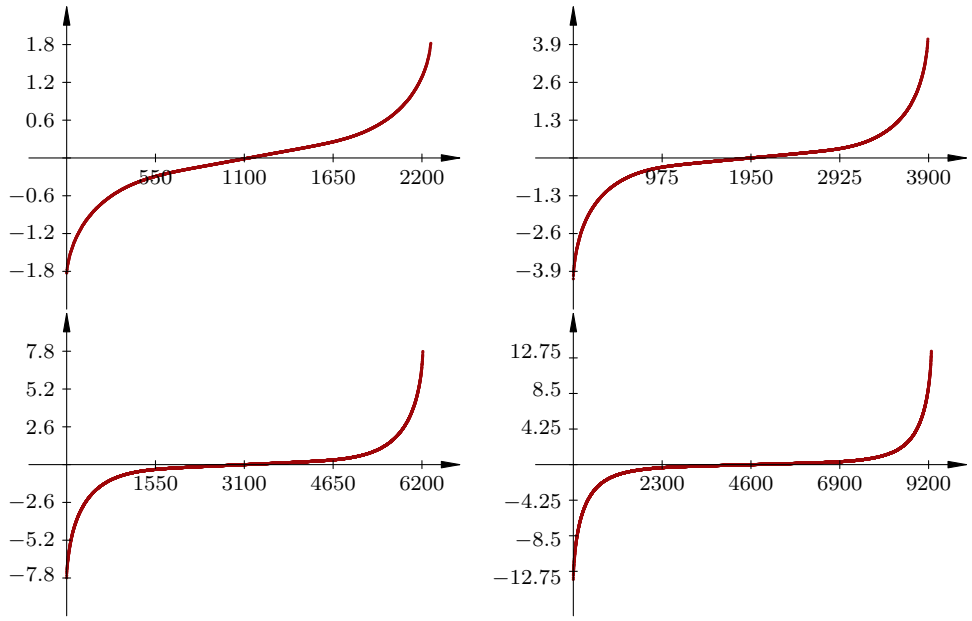

FIG. S.3. *Left to right, top to bottom: the generalized eigenvalues* $\Lambda(B1)$, $\Lambda(B2)$, $\Lambda(B3)$, $\Lambda(B4)$.

and from the `ZHEGVD` routine in the right subfigures, in all cases with 64 threads. Notice that the above expression preserves the directionality of the relative errors.

Table S.2 is an addendum to subsection 6.5.4 of the main paper. The results with *24* threads were obtained on [1], and those with 32 threads on an Intel Xeon Phi 7210.

TABLE S.2

The wall times for the explicit formation of H *and* S*, combined with those for a LAPACK's generalized Hermitian eigensolver (`ZHEGV` or `ZHEGVD`), and the speedup vs.* $(\star)$*, with 32 and* 24 *threads.*

ID	# of thrs.	(max. of 2 runs for H, S) wall time [s] for $H = \widetilde{F}^* J \widetilde{F}$	$S = \widetilde{G}^* \widetilde{G}$	`ZHEGV`	`ZHEGVD`	total wall time [s] with `ZHEGV` $(\triangleleft)$	`ZHEGVD` $(\triangleright)$	speedup vs. $(\star)$ $(\star)/(\triangleleft)$	$(\star)/(\triangleright)$
A1	32	2.441	2.003	13.229	5.800	17.673	10.087	16.902	29.612
	24	1.709	0.720	3.211	1.889	5.640	3.863	20.588	30.055
A2	32	7.175	4.432	44.349	24.058	55.925	35.665	19.647	30.809
	24	3.599	1.951	13.933	10.106	19.467	15.628	21.480	26.756
A3	32	18.039	10.953	150.096	82.914	179.037	111.895	19.309	30.896
	24	8.842	4.761	59.503	41.403	73.105	54.991	20.001	26.589
A4	32	40.471	23.477	404.234	242.245	467.785	306.193	21.775	33.267
	24	19.376	10.557	190.010	123.832	219.942	153.662	20.129	28.812
B1	32	2.254	2.201	5.855	2.310	10.310	6.592	23.234	36.341
	24	1.249	0.769	1.098	0.636	3.051	2.643	29.530	34.091
B2	32	7.340	4.343	20.335	8.764	31.754	20.447	22.836	35.465
	24	3.381	1.885	4.817	3.220	9.973	8.486	27.240	32.015
B3	32	18.556	11.224	59.596	31.903	87.790	61.683	23.819	33.900
	24	8.246	4.548	18.200	13.508	30.995	26.166	24.355	28.849
B4	32	38.886	22.372	162.476	91.045	223.733	151.706	23.217	34.240
	24	18.096	10.175	55.711	43.620	83.982	71.541	23.860	28.009

S.3. Addendum to section 7.

S.3.1. Computing the relative errors. The relative errors from (7.1) have to be obtained in a higher precision that the one used for the rest of the computation, to reduce the effects of the rounding errors. However, the exact (or "infinite" precision) arithmetic is prohibitively expensive with the data of all but very small dimensions.

That implies using either the 128-bit quadruple (emulated in software and thus slower) or the 80-bit Intel extended (hardware provided) floating-point types. We chose the latter, but it is supported only by the GNU Fortran compiler with `KIND=10`.

The data is read in double precision and promoted to extended precision. The column scalings by Σ_F and Σ_G, the matrix multiplications, and the Frobenius norm computation are manually parallelized with OpenMP. The last operation is performed by accumulating $\bar{z} \cdot z$ per each column (followed by a reduction and a square root), but since the exponent range of the datatype is also significantly wider than what is contained in the data, no overflow or underflow should occur.

Accuracy results. In Table S.3 the relative errors for the full dataset are shown.

TABLE S.3

The relative errors in the full GHSVD (from the Phases 3 and 4, with 32 and 64 threads), obtained in extended precision. While $\|F\|_F$ does not vary with the number of threads used in Phase 2, $\|G\|_F$ varies so negligibly that the rounding to the six decimal places shown is not affected.

ID	$\|F\|_F$	$\|G\|_F$	$\|F - U\Sigma_F X\|_F/\|F\|_F$		$\|G - V\Sigma_G X\|_F/\|G\|_F$	
			32 threads	64 threads	32 threads	64 threads
A1	197.339487	57.118214	1.250533e−13	1.260103e−13	9.474358e−14	9.644613e−14
A2	318.701344	75.513862	1.618982e−13	1.609144e−13	2.064587e−13	2.069223e−13
A3	317.883120	85.813833	2.721798e−13	2.716406e−13	4.533952e−13	4.530720e−13
A4	385.297523	103.571436	3.510818e−13	3.517243e−13	8.234634e−13	8.234643e−13
B1	148.478058	40.515148	1.120428e−13	1.123317e−13	7.981249e−14	7.976024e−14
B2	275.148879	54.161577	2.244659e−13	2.234393e−13	1.569443e−13	1.566841e−13
B3	423.840558	66.447550	4.175919e−13	4.196549e−13	2.821371e−13	2.813644e−13
B4	565.824887	76.845006	7.181818e−13	7.223445e−13	4.731078e−13	4.733831e−13

An alternative definition of the relative errors. Another option to compute the relative errors is to look at $\|FZ - U\Sigma_F\|_F/\|F\|_F$ and $\|GZ - V\Sigma_G\|_F/\|G\|_F$, thus avoiding Phase 4, but is neither general enough to be used for the other types of the G(H)SVD algorithms, not it would account for the effects of Phase 4, should it be employed after the execution of the algorithms described in the main paper.

For experimenting with different combinations of precision of arithmetic and in-memory data, up to quadruple precision of either or both, a separate software distribution is also freely available in https://github.com/venovako/MPHZ repository.

S.3.2. Comparison with `ZGGSVD3`. In Table S.4 the speedup of the Phase 3 followed by the Phase 4 (both with 32 threads) versus `ZGGSVD3` is shown. Neither here nor in Table 7.1 in the main paper the timings include the block column reordering, necessary between the Phases 3 and 4, from the order given by the first step of a parallel Jacobi strategy to the natural one, with the ascending block indices. However, that would amount to a permutation by copying (similarly to forming $\widetilde{G}P_2$ in subsection 5.4.1 of the main paper in a combined implementation of those phases, and from Table 5.1 in the main paper it follows that such reshuffling would have a

TABLE S.4

The wall execution time (wtime) and the speedup of the Phases 3 (with $J = I$) and 4 versus the `ZGGSVD3` *LAPACK routine on the set C, with 32 threads and n denoting the order of the matrices.*

n	`ZGGSVD3` wtime [s] (•)	Phase 3 wtime [s] & sweeps	Phase 4 wtime [s]	Phases 3 & 4 wtime [s] (○)	speedup (•)/(○)
1000	367.13	6.10; 13	3.14	9.24	39.74
2000	5873.70	33.15; 14	22.06	55.21	106.40
3000	21991.07	100.28; 16	71.58	171.86	127.96
4000	54233.01	228.10; 17	199.02	427.11	126.98
5000	107167.78	436.18; 17	325.85	762.04	140.63

negligible impact on the overall speedup for the matrices large enough.

REFERENCES

[1] JÜLICH SUPERCOMPUTING CENTRE, *JUWELS: Modular Tier-0/1 Supercomputer at the Jülich Supercomputing Centre*, Journal of large-scale research facilities, 5, A135 (2019), https://doi.org/10.17815/jlsrf-5-171.

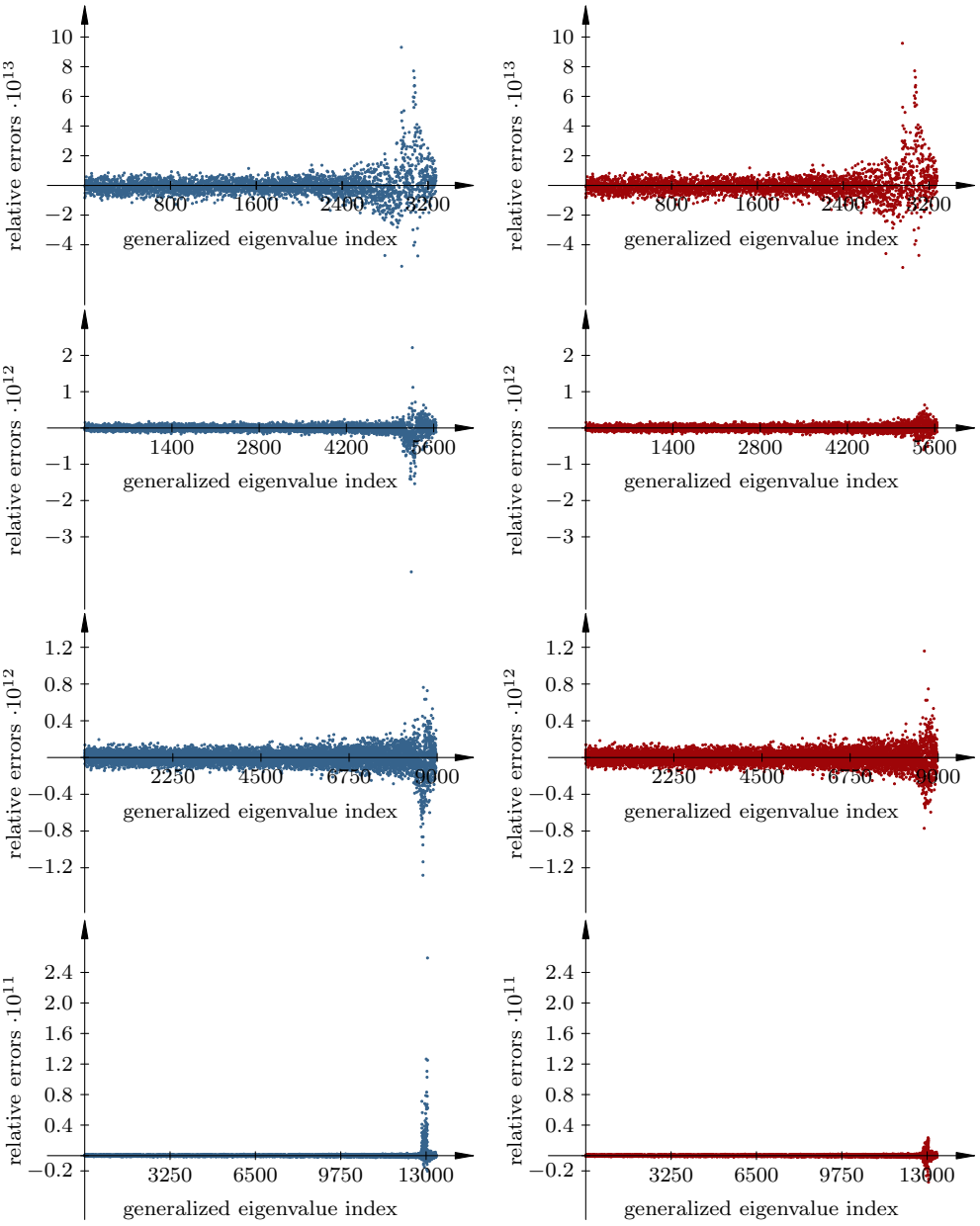

FIG. S.4. *From top to bottom: the errors in the generalized eigenvalues* $\Lambda(A1)$, $\Lambda(A2)$, $\Lambda(A3)$, $\Lambda(A4)$, *from the LAPACK routines* **`ZHEGV`** *(left & blue) and* **`ZHEGVD`** *(right & red), relative to the generalized eigenvalues computed by Phase 2 and then the Level 2 (BO) Phase 3 GHSVD.*

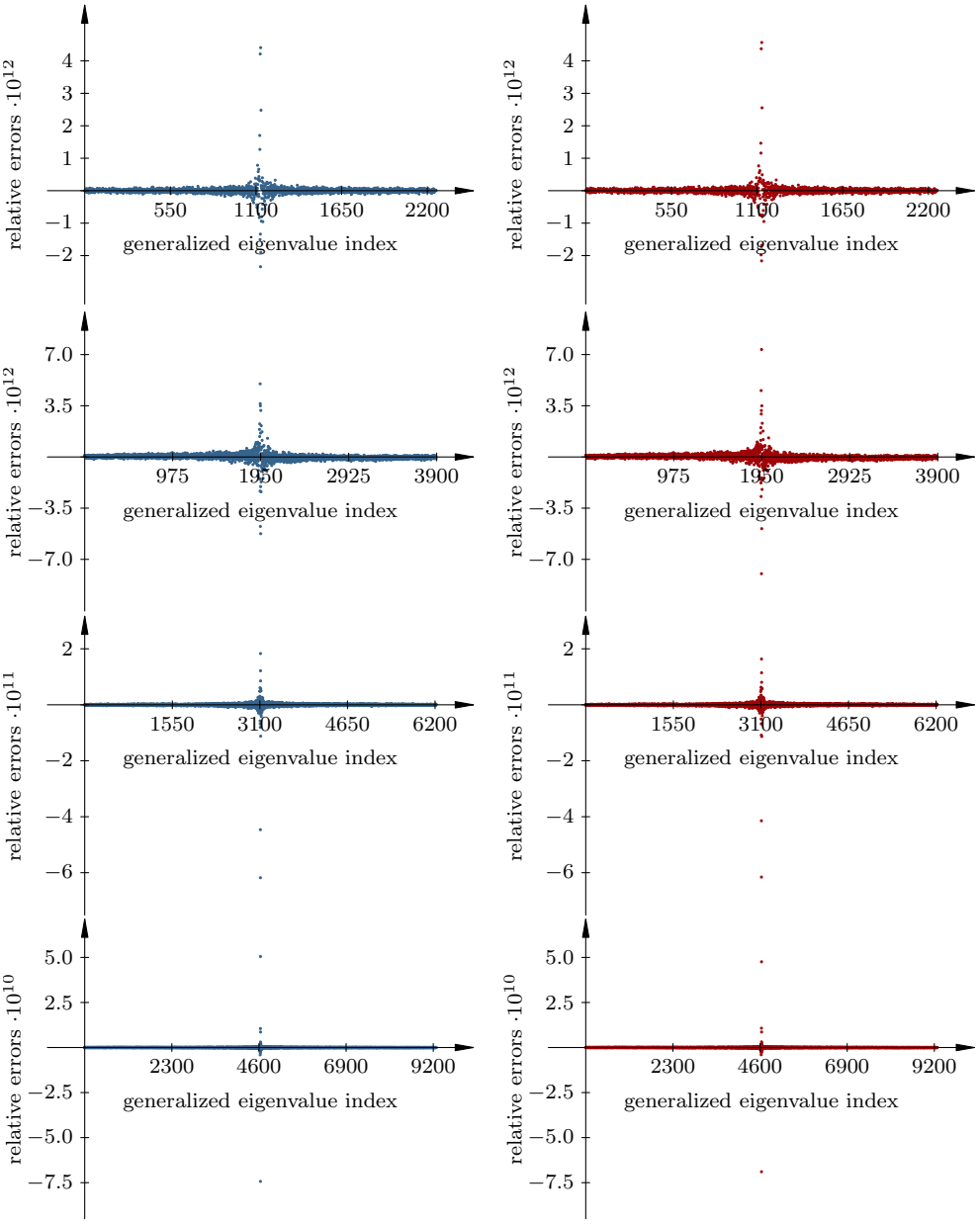

FIG. S.5. *From top to bottom: the errors in the generalized eigenvalues* $\Lambda(B1)$, $\Lambda(B2)$, $\Lambda(B3)$, $\Lambda(B4)$, *from the LAPACK routines* **`ZHEGV`** *(left & blue) and* **`ZHEGVD`** *(right & red), relative to the generalized eigenvalues computed by Phase 2 and then the Level 2 (BO) Phase 3 GHSVD.*