Collaboration between computational material science and applied mathematics on supercomputer: numerical libraries and their algorithms

Takeo Hoshi (Tottori U.)

1. Overview
2. EigenKernel
3. Kω
4. k-ep

Self-introduction: Takeo Hoshi (Tottori U)

Research area: computational condensed matter physics, applied mathematic

Fig. Massively parallel electronic state calculation with *ab-initio* based modelled (TB) theory
(ELSES (http://www.elses.jp); T. Hoshi, Proc. ScalA16 in SC16, pp.33-40 (2016);)

Fig. (a) Strong scaling bench mark of 10^8-atom condensed organic polymers on the K computer

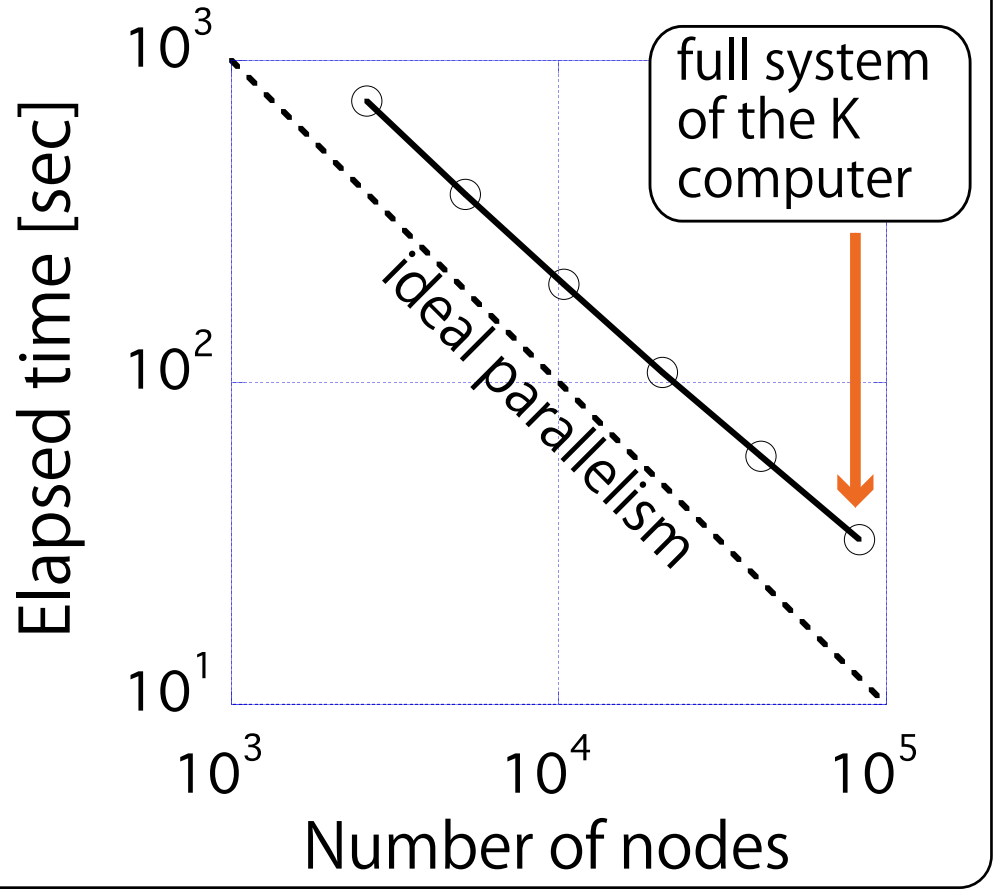
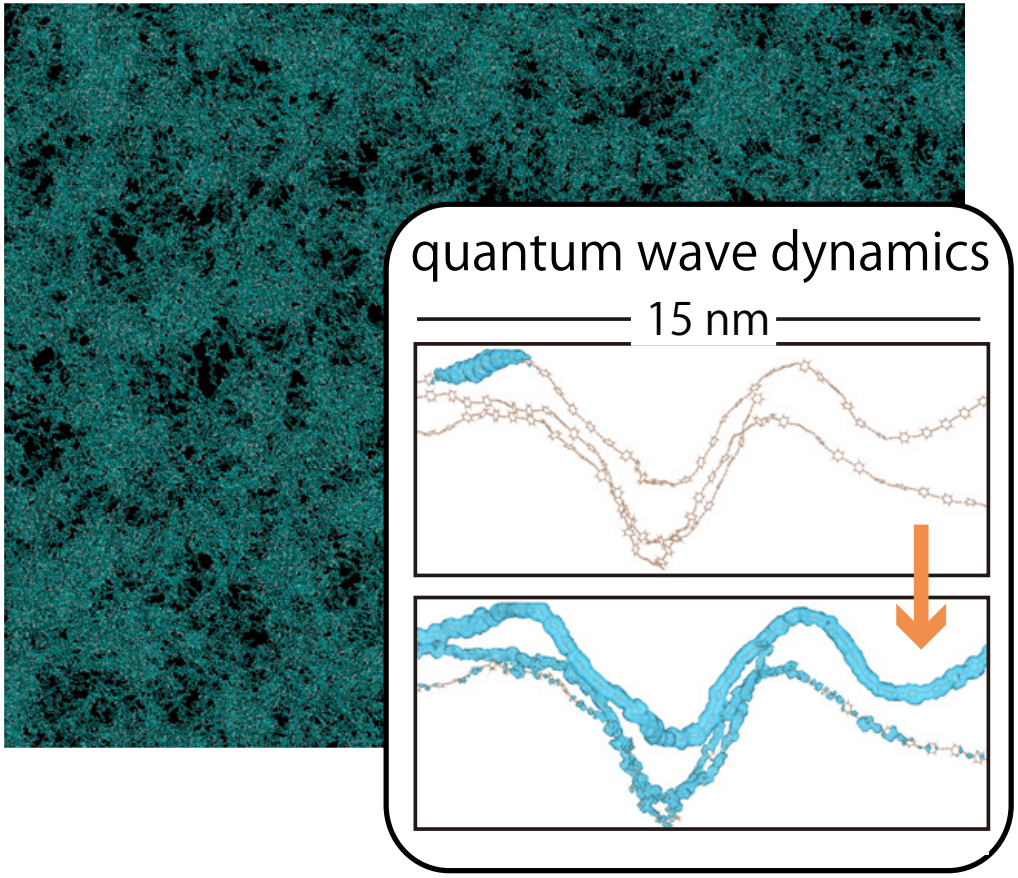


Fig. (b) Research example: 10^8-atom (100nm-scale) condensed organic polymers for electronic transport mechanism



Overview of numerical linear algebra with large matrices

What is 'large matrix' ?

-> the size of M >> 10^4

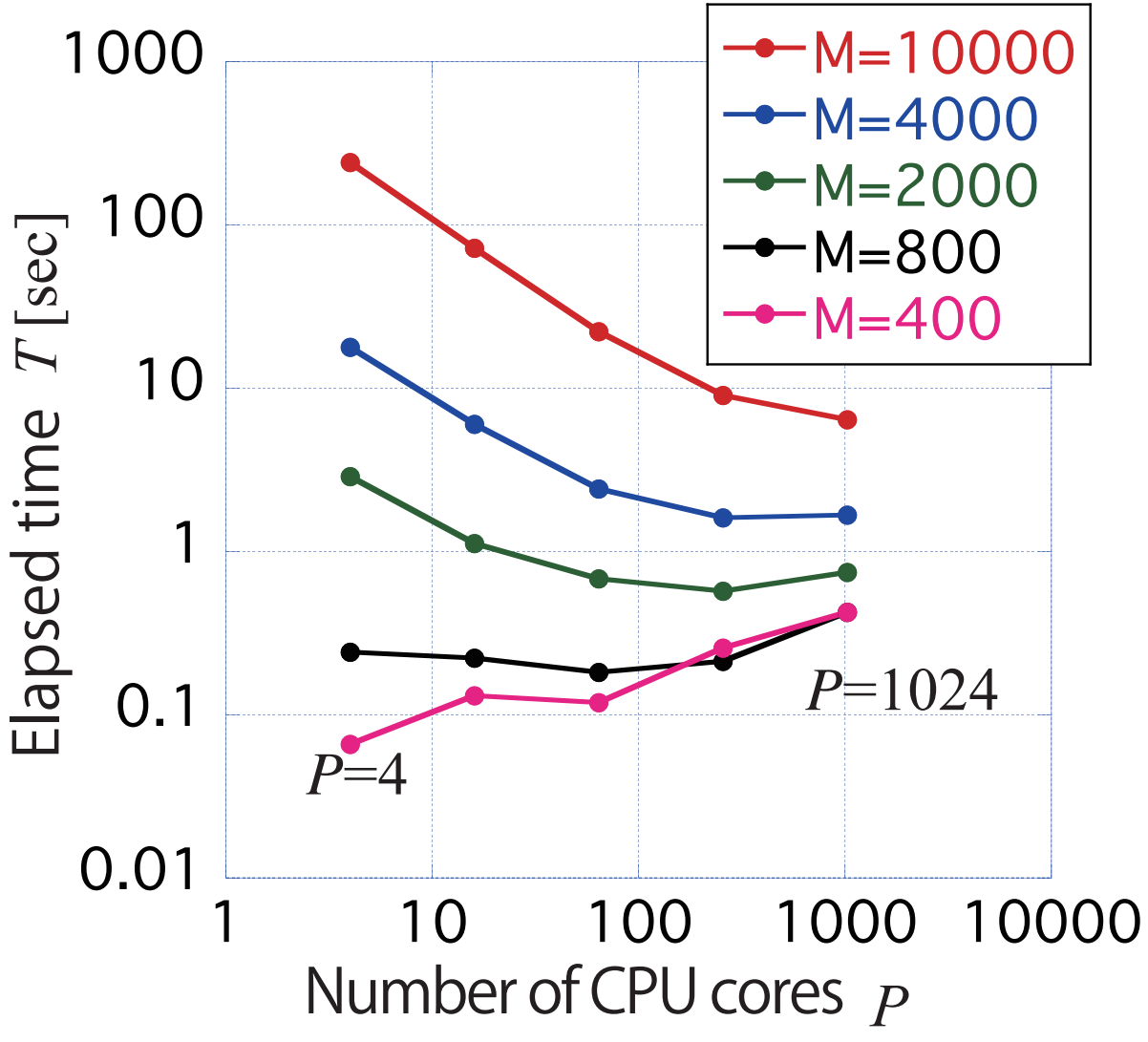
(a) $M=10^3$: small size
(required memory size)
$= 8\ B \times M^2 = 8MB$

(b) $M=10^4$: middle size
(required memory size)
$= 8\ B \times M^2 = 0.8GB$

(c) $M=10^5$: large size
(required memory size)
$= 8\ B \times M^2 = 80GB$

Note: built-in memory size
on one node in Sekirei
(ISSP supercomputer):
= 128 GB ('CPU node')
or 1TB ('Fat node')

Ex. Eigenvalue problem
with ScaLAPACK



(*) Intel Xeon (SGI Altix @ISSP), real symmetric mat.

Classification of mathematical concepts in solvers

	dense or sparse	exact or approx.	library example	today's topic
Direct solver	dense	exact	LAPACK ScaLAPACK	EigenKernel
Sparse-direct solver	sparse	exact	PARDISO (in Intel MKL) MUMPS	k-ep
Krylov-subspace solver	sparse	approx.	PETSc	Kω

Strategies of matrix solvers

Krylov (iterative) solvers
- ‘projection’ strategy
- (mainly) sparse matrices

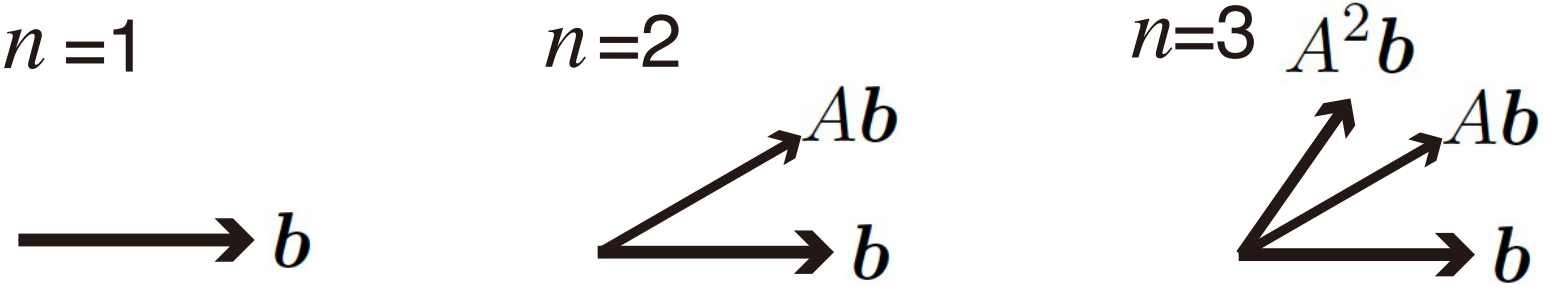
Direct solvers
- ‘transformation’ strategy
- (mainly) dense matrices
- O(N^3) calculation

ex. CG algorithm

‘projection’ on the Krylov subspace of

$$K_n(A;\boldsymbol{b}) \equiv \mathrm{span}\left\{\boldsymbol{b}, A\boldsymbol{b}, A^2\boldsymbol{b}, ..., A^{n-1}\boldsymbol{b}\right\}$$

n : subspace dimension (iteration number)



Strategies of matrix solvers

Krylov (iterative) solvers
- ‘projection’ strategy
- (mainly) sparse matrices

Direct solvers

ex. CG algorithm for $A\boldsymbol{x} = \boldsymbol{b}$
→ the n-th step solution $\boldsymbol{x}_n$ is given by

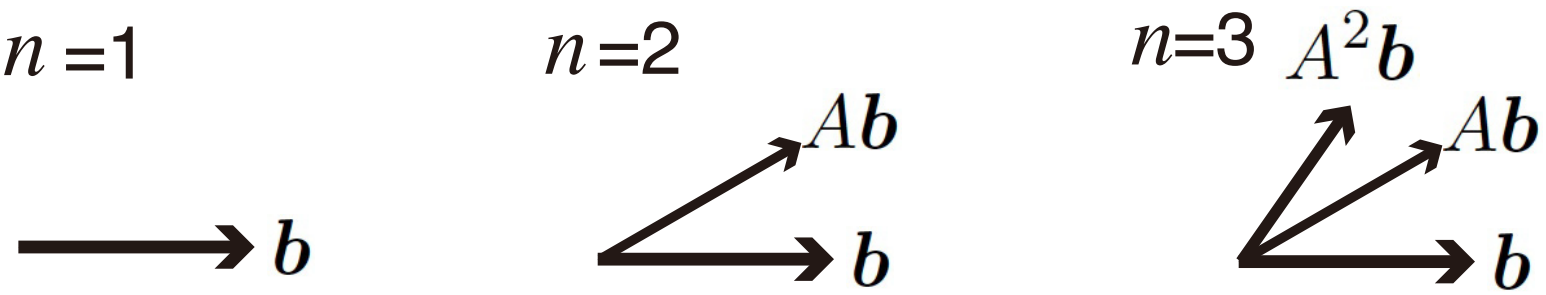
$$\boldsymbol{x}_n \in K_n(A; \boldsymbol{b})$$

ex. CG algorithm

‘projection’ on the Krylov subspace of

$$K_n(A; \boldsymbol{b}) \equiv \mathrm{span}\left\{\boldsymbol{b}, A\boldsymbol{b}, A^2\boldsymbol{b}, ..., A^{n-1}\boldsymbol{b}\right\}$$

n : subspace dimension (iteration number)



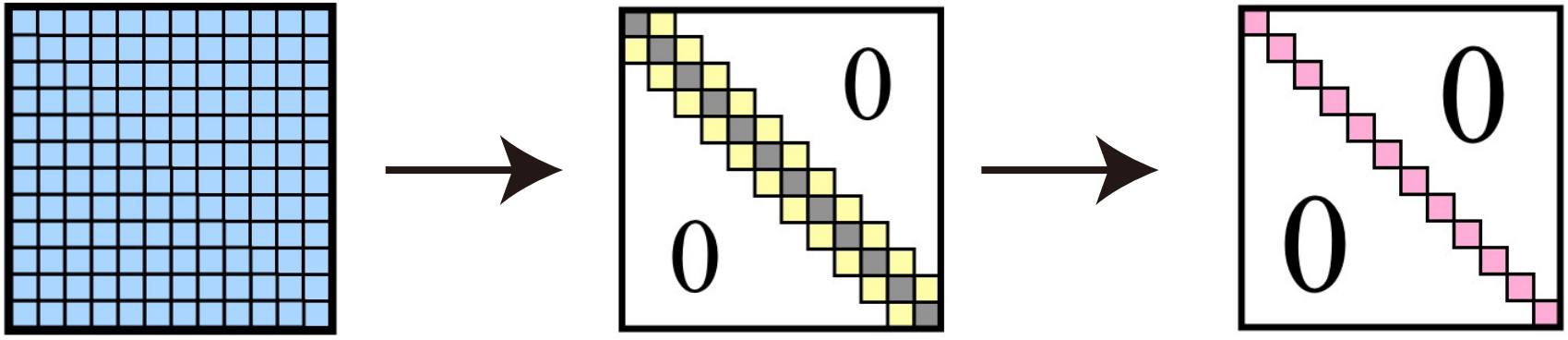
Strategies of matrix solvers

Krylov (iterative) solvers
- ‘projection’ strategy
- (mainly) sparse matrices

Direct solvers
- ‘transformation’ strategy
- (mainly) dense matrices
- O(N^3) calculation

‘transformation’ of the whole space

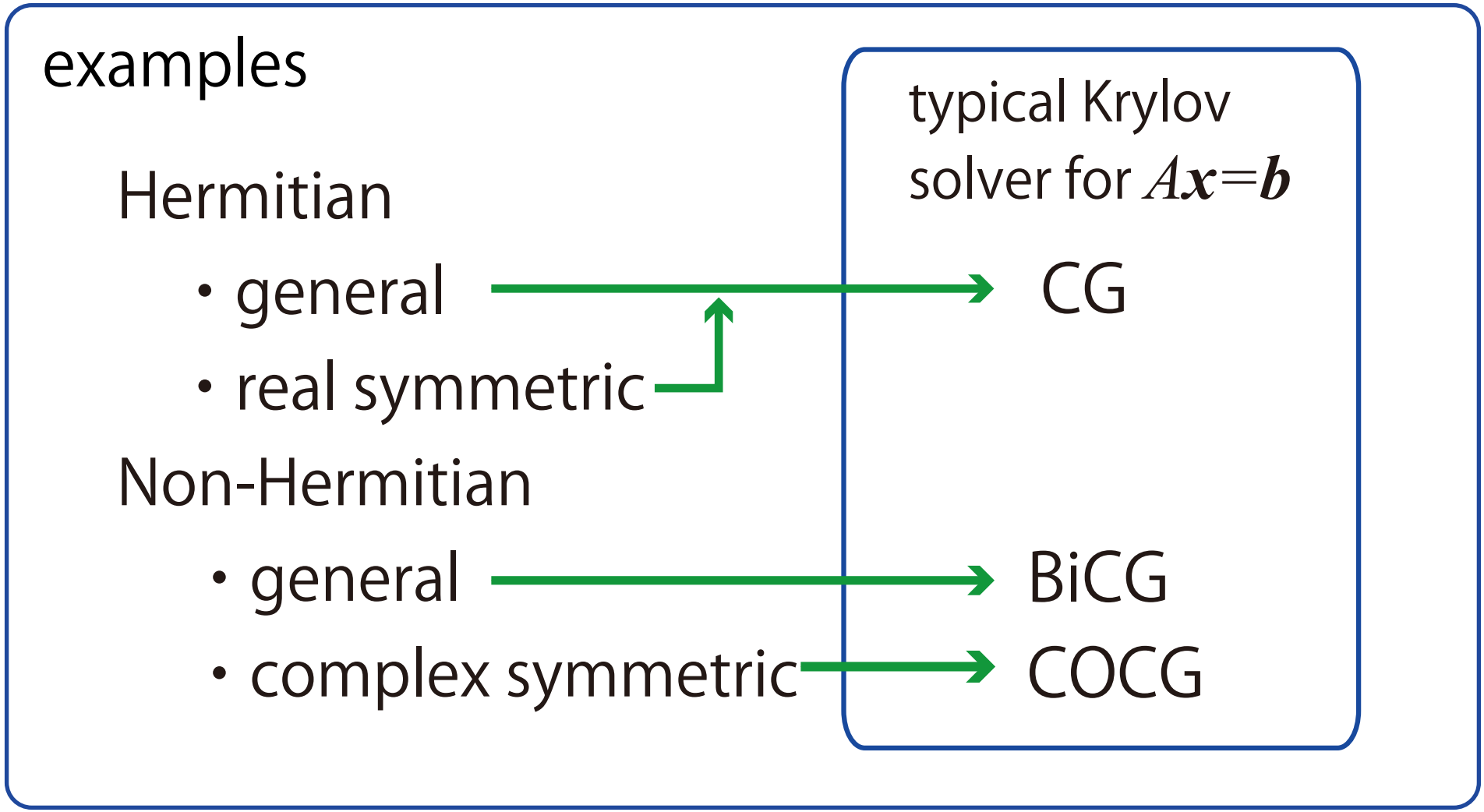
Example in eigenvalue problem



Classification of mathematical concepts in solvers

	dense or sparse	exact or approx.	library example	today's topic
Direct solver	dense	exact	LAPACK ScaLAPACK	EigenKernel
Sparse-direct solver	sparse	exact	PARDISO (in Intel MKL) MUMPS	k-ep
Krylov-subspace solver	sparse	approx.	PETSc	Kω

Classification of solver by matrix type



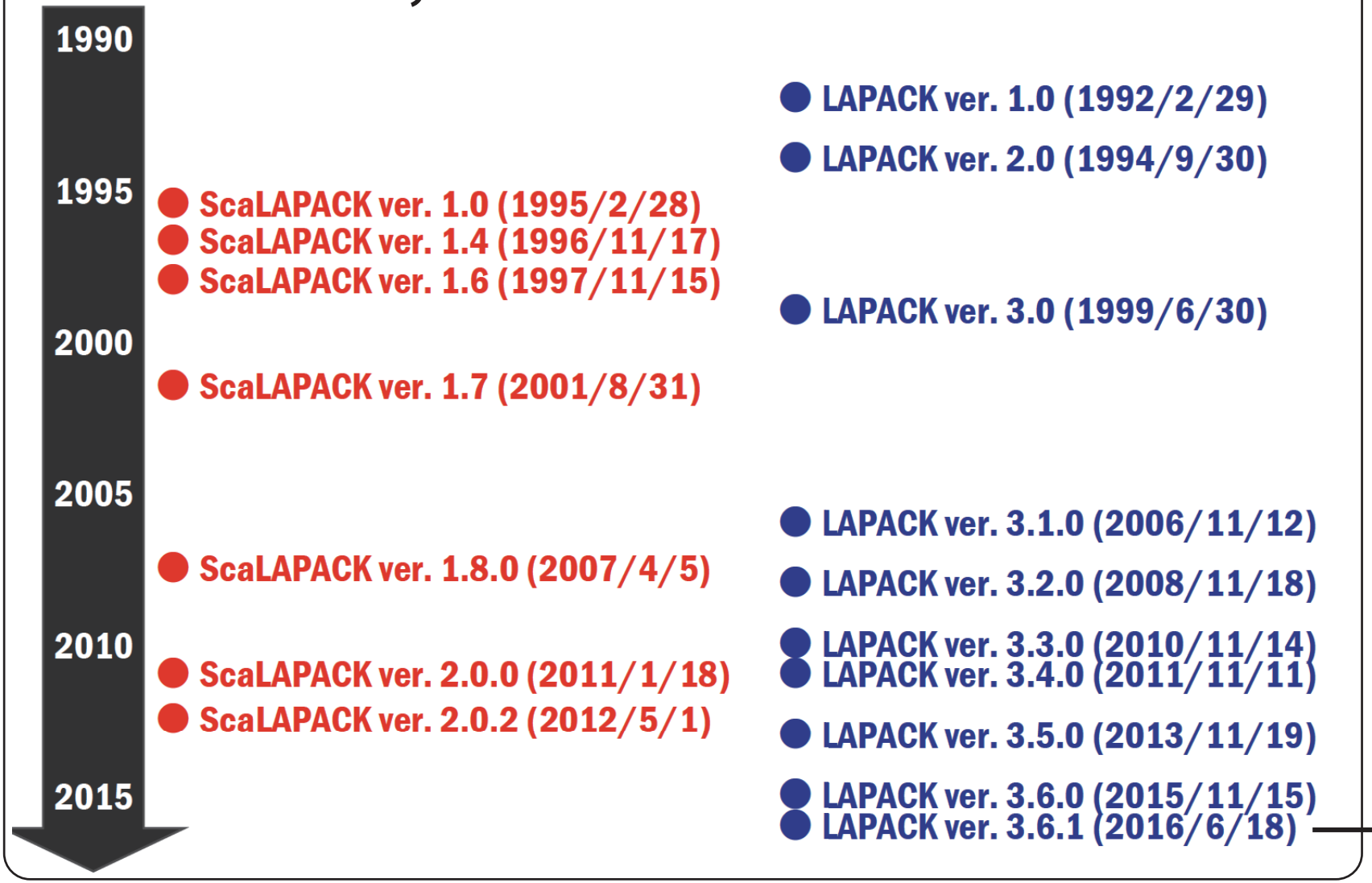
Classification of mathematical concepts in solvers

	dense or sparse	exact or approx.	library example	today's topic
Direct solver	dense	exact	LAPACK ScaLAPACK	EigenKernel
Sparse-direct solver	sparse	exact	PARDISO (in Intel MKL) MUMPS	k-ep
Krylov-subspace solver	sparse	approx.	PETSc	Kω

ScaLAPACK: Scalable Linear Algebra PACKage (1/2)

- distributed parallel routines based on LAPACK(LinearAlgebraPACKage)
- de facto standard but ....

Version history



from Fukaya (Hokkaido U)'s slide (2016) in https://sites.google.com/site/upscaling2016/

ScaLAPACK: Scalable Linear Algebra PACKage (2/2)

- distributed parallel routines based on LAPACK(LinearAlgebraPACKage)
- de facto standard but ....

(partial) comparison between LAPACK and ScaLAPACK

Problem	LAPACK	ScaLAPACK
Linear Equation	GESV (LU)	PxGESV
	POSV (Cholesky)	PxPOSV
	SYSV (LDL^T)	missing
Least Squares	GELS (QR)	PxGELS
	GELSY (QR with pivoting)	missing driver
Symmetric EVD	SYEV (inverse iteration)	PxSYEV
	SYEVD (D&C)	missing
	SYEVR (RRR)	missing
Nonsymmetric EVD	GEES (HQR)	missing driver
	GEEV (HQR + vectors)	missing driver
SVD	GESVD (QR)	PxGESVD
	GESDD (D&C)	missing

from Fukaya (Hokkaido U)'s slide (2016) in https://sites.google.com/site/upscaling2016/

Sparse-matrix data files in matrix-market format

```
%%MatrixMarket matrix coordinate real general
%
5  5  8
   1    1      1.0
   2    2     10.5
   3    3    0.015
   1    4      6.0
   4    2     250.5
   4    4    -280.0
   4    5     33.32
   5    5     12.0
```

file example: a 5x5 matrix,
(num. of non-zero elem.)=8
matrix data

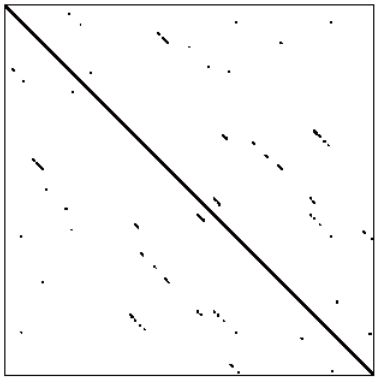
```
1     0      0      6      0
0    10.5    0      0      0
0     0     .015    0      0
0   250.5    0     -280   33.32
0     0      0      0      12
```


[1] Matrix Market
https://math.nist.gov/MatrixMarket/

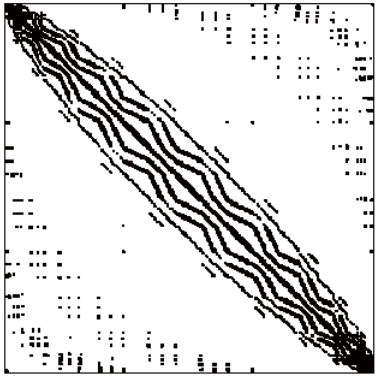
[2] The SuiteSparse Matrix Collection
(formerly the University of Florida
Sparse Matrix Collection)
https://sparse.tamu.edu/

[3] ELSES matrix library (our own)
http://www.elses.jp/matrix/

Example of visualized non-zero elements [3]



CPPE32346

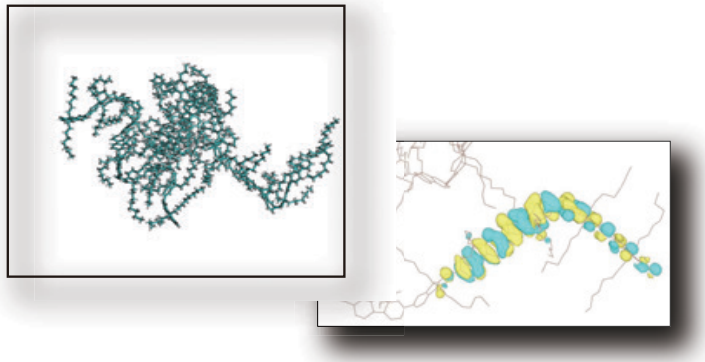


NCCS430080

Application-Algotirhm-Architecture co-design

A common foundation for post-K (and post-Moore) era

Application : Electronic state calculation (or any field)

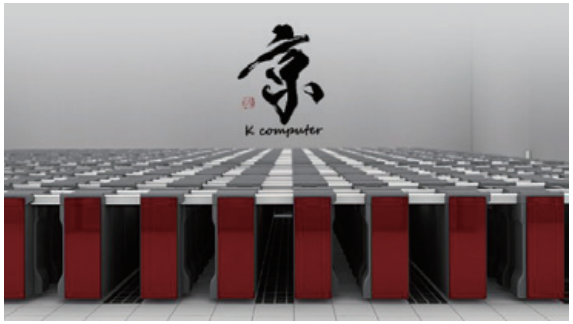


Algorithm : Numerical linear algebra

$$Hy = \lambda Sy$$

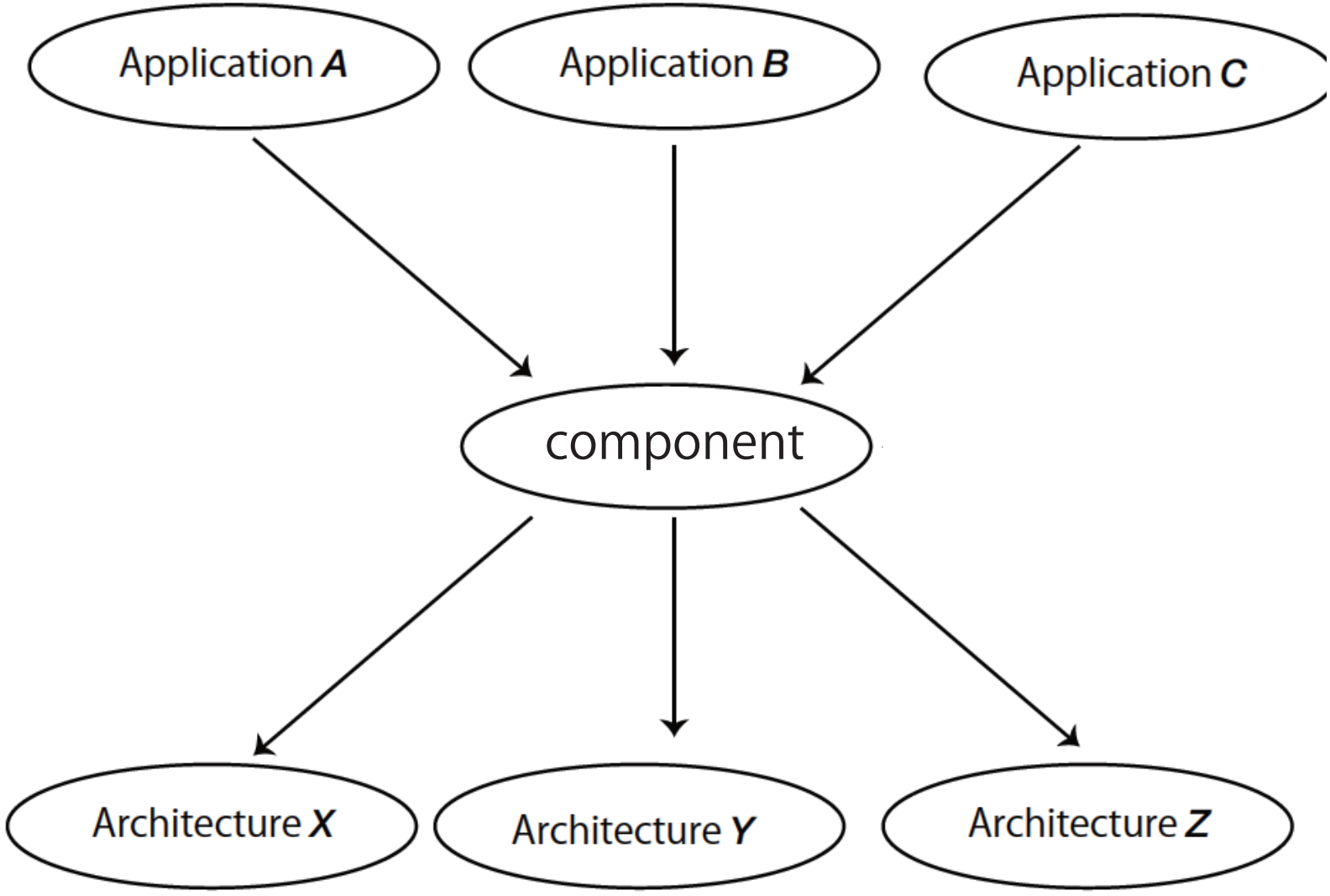
$$(zS - H)x = b$$

Architecture : The K computer (and beyond)



Application-Algotirhm-Architecture co-design

Key issue: sharing mathematical components or ‘building block’



ELPA: Eigenvalue SoLvers for Petaflop-Applications

http://elpa.rzg.mpg.de/

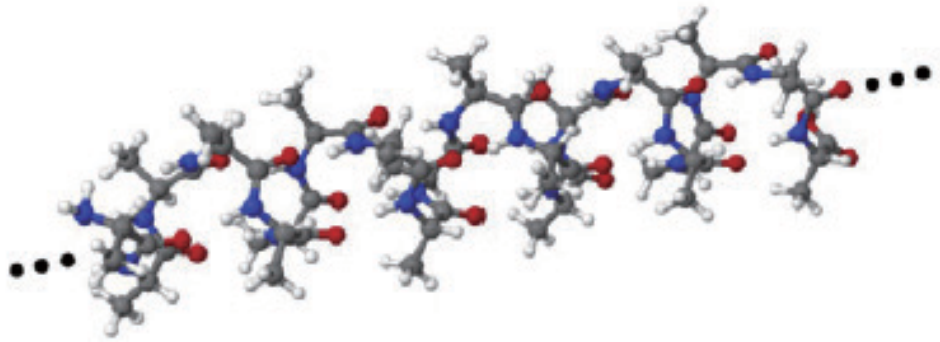
- Novel numerical solver of generalized eigenvalue problem
- Developed in the tight-collaboration
  with electronic-state calculation community in Europe

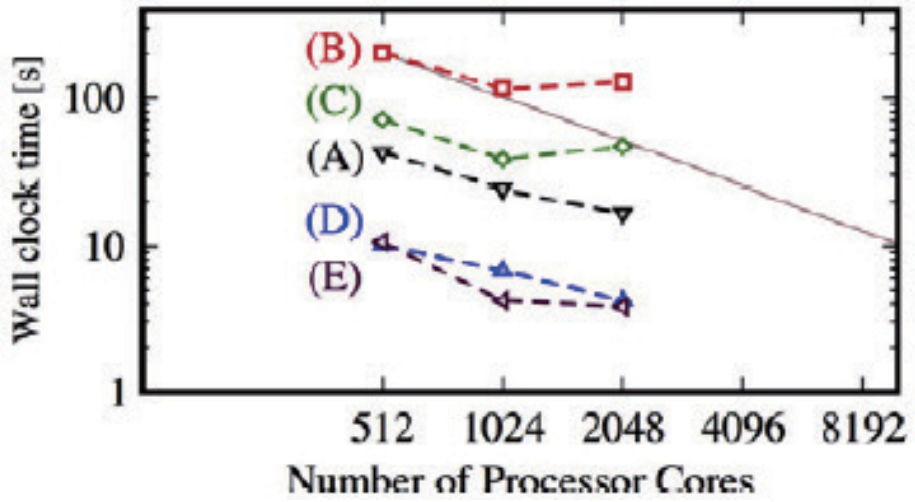
[1] T. Auckenthaler et al., Parallel Comput. 37, 783 (2011)

[2] A. Marek et al., J. Phys.: Condens. Matter 26, 213201(2014).

[3] https://arxiv.org/abs/1806.01036

ex：α-helical polyalanine molecule [1]





Classification of mathematical concepts in solvers

	dense or sparse	exact or approx.	library example	today's topic
Direct solver	dense	exact	LAPACK ScaLAPACK	EigenKernel
Sparse-direct solver	sparse	exact	PARDISO (in Intel MKL) MUMPS	k-ep
Krylov-subspace solver	sparse	approx.	PETSc	Kω

Several advanced topics

1. Performance prediction (and auto-optimization)
  ・Our work (Bayesian inference with Monte Carlo sampling)
    Tanaka et al., https://arxiv.org/abs/1806.00741

2. Matrix function (→ Prof. Sogabe's talk ?)

3. Numerical computation method in the data-driven science
  ・Note: Architectures for deep learning are designed so as to
    enhance the performance in single or half precision computation.
  → hopeful idea: Error-controlled mixed precision calculation
    ((numerical error) < (statistical error))
  ・Collaboration with Prof. Ogita (TWCU, Post-K proj.)
    for verified numerical computations
    (T. Hoshi, in SCAN2018, Waseda U, Sep. 10-15, 2018)
  cf. (Mixed precision in ELPA) https://arxiv.org/abs/1806.01036

EigenKernel :
a middleware-type eigenvalue solver

EigenKernel : a middleware-type library

- A fundamental numerical problem in electronic state calculations is generalized eigenvalue problem (GEP)

$$A\boldsymbol{y}_k = \lambda_k B\boldsymbol{y}_k \qquad (1)$$

A : real symmetric

B : real symmetric, positive definite

- The dense-matrix solver for GEP is reduced to the standard eigenvalue problem (SEP)

$$A'\boldsymbol{z}_k = \lambda_k \boldsymbol{z}_k \qquad (2)$$

where the 'reducer' routines give

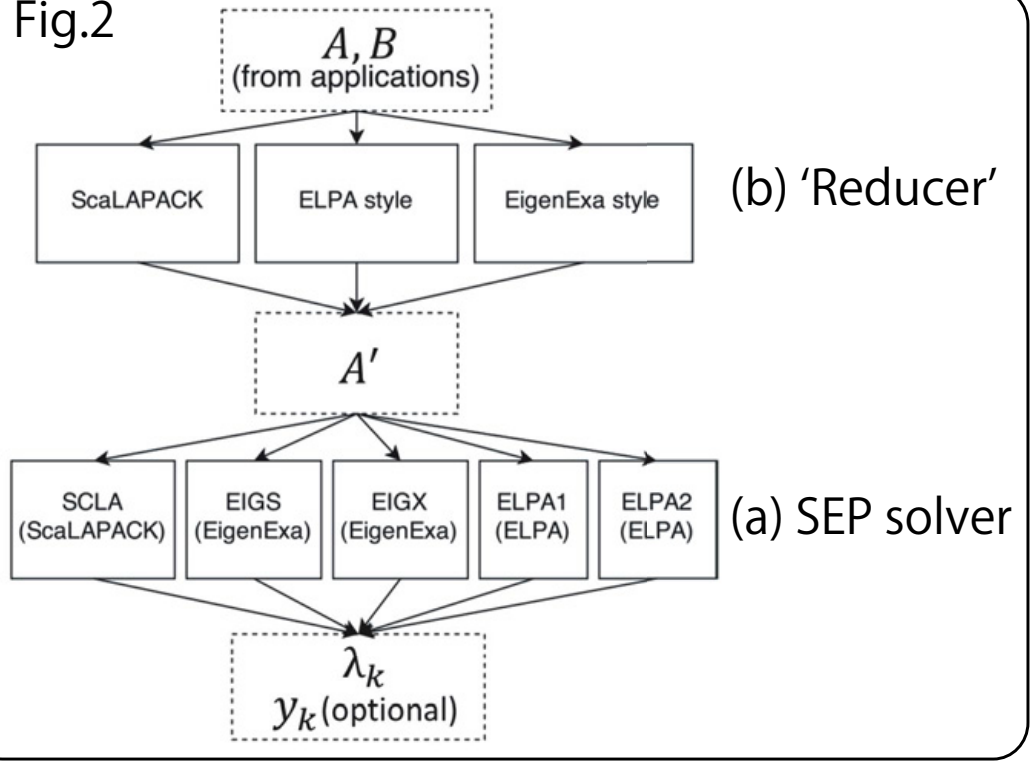
$$U := \mathrm{chol}(B) \qquad (3)$$

$$(B = U^{\mathrm{T}}U \qquad (4)\ )$$

$$A' := U^{-\mathrm{T}}AU^{-1} \qquad (5)$$

- EigenKernel enables a hybrid solver by choosing
  (a) SEP solver for Eq.(2)
  (b) 'Reducer' between GEP and SEP, ex. Eqs.(3),(5)
  from ScaLAPACK, ELPA and EigenExa routines, as in Fig. 2

Fig.2



[0] https://github.com/eigenkernel/
[1] H. Imachi and T. Hoshi, J. Inf. Process. 24, 164 -172 (2016)
[2] https://arxiv.org/abs/1806.00741

EigenKernel : a middleware-type library

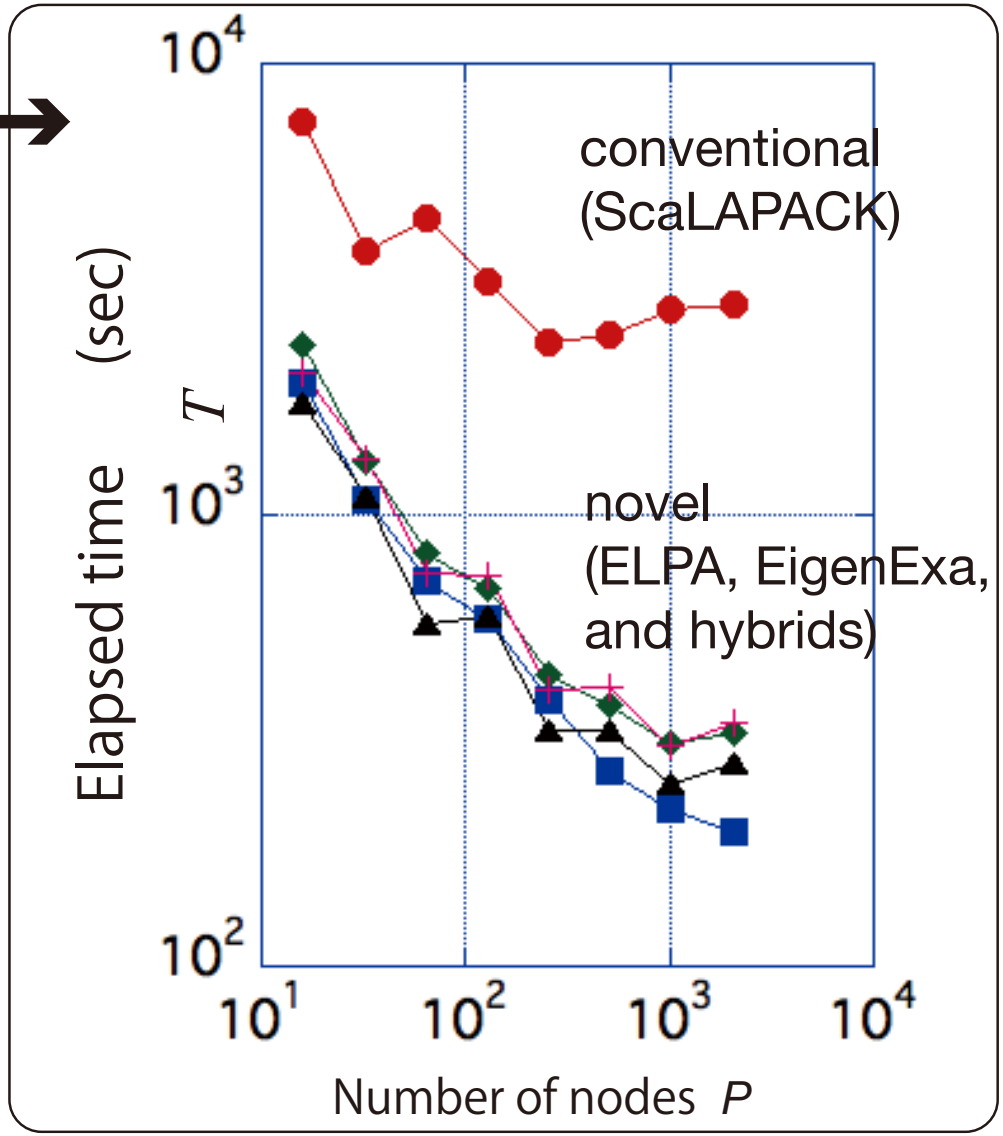
(a) The maximum matrix size is $M=10^6$ on the full system of the K computer within 1.5 hours. (The world record)

T. Hoshi et al., Proc. ScalA16 in SC16, 33-40, (2016)

(b) Benchmark on Oakforest-PACS

- Matrix size : M = 90,000
- Number of processor nodes
  P = 1 ~ 2048 (a quater of OFP)

https://arxiv.org/abs/1806.00741



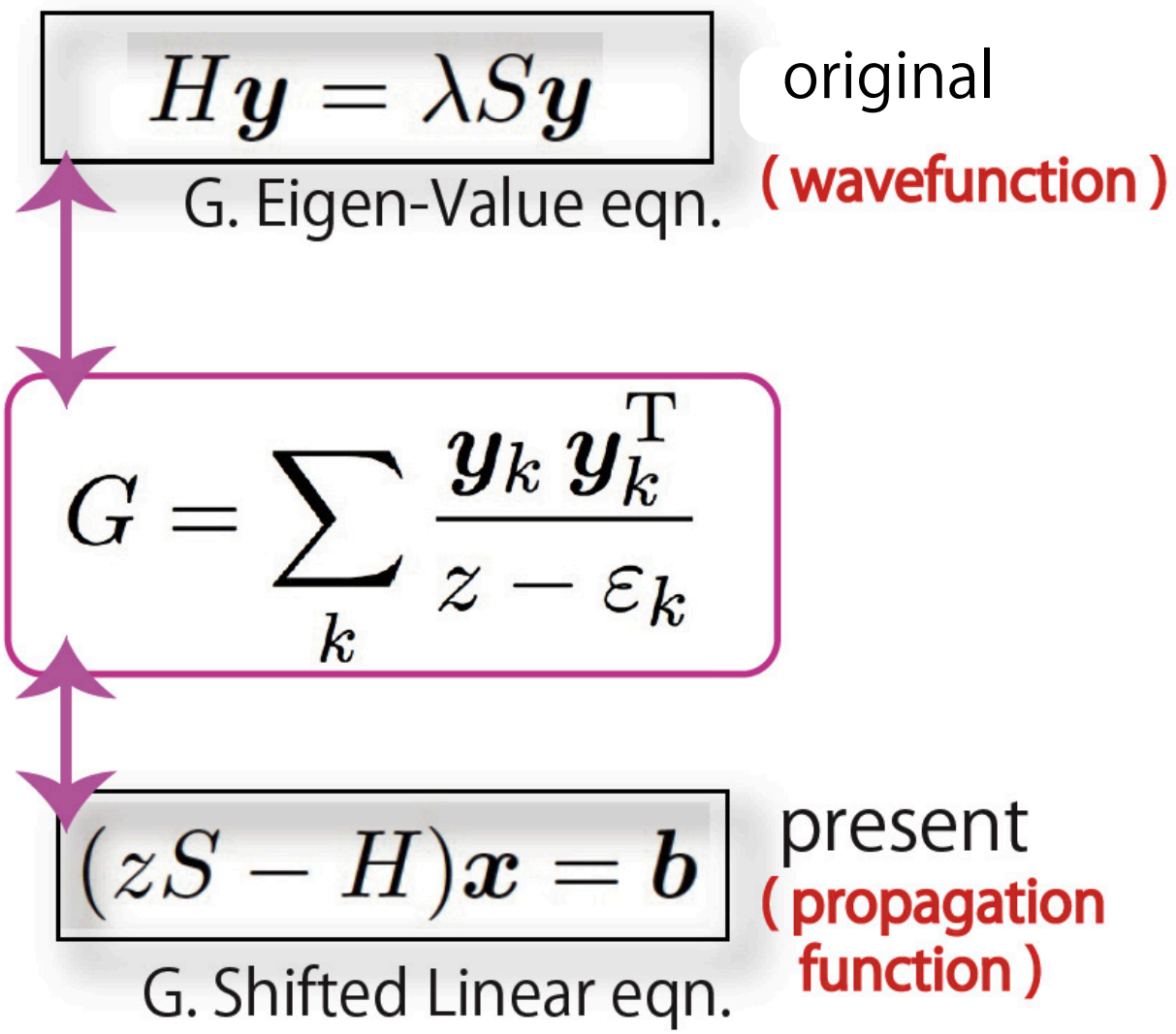
Possible test calculation

- Use EigenKernel with various solver algorithms
- Test matrix data can be found in
  ELSES matrix library (http://www.elses.jp/matrix/)
- Detailed computational times are written in an output file
- The same kind of test
  can be carried out also with k-ep

Kω :

shifted Krylov-subspace solver

Appearance of (generalized) shifted linear equations



Kω：solver of shifted Krylov-subspace method

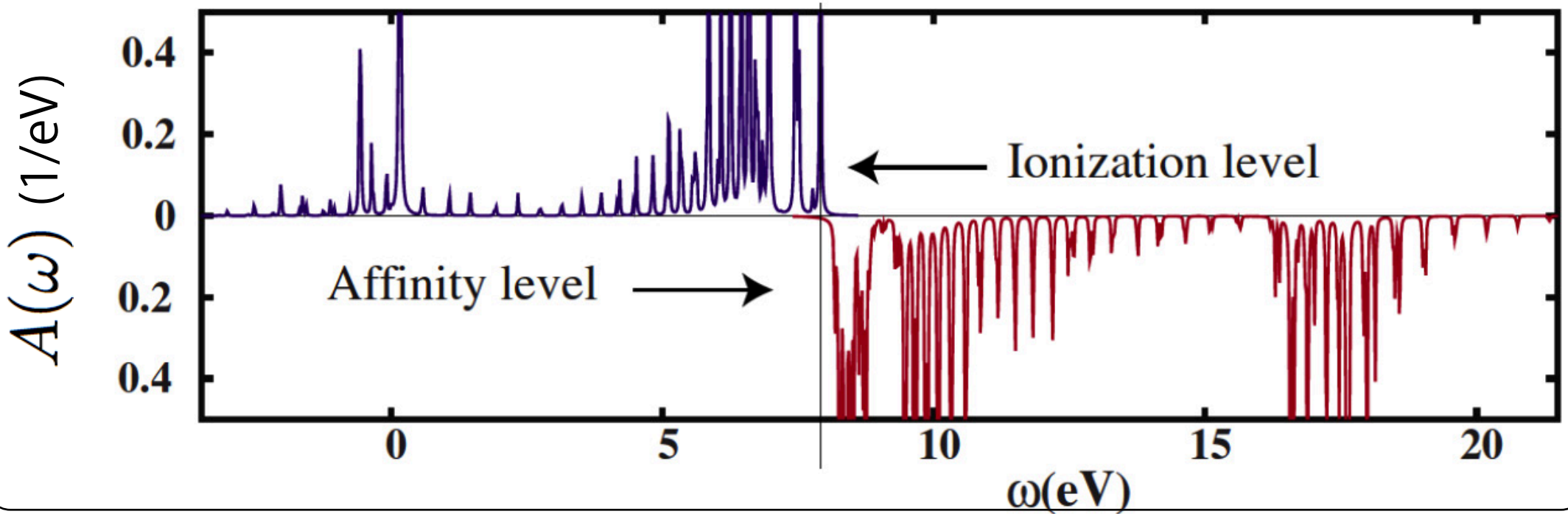
(one of)target: excited spectrum calculation in quantum lattice model

$$A(\omega) = -\frac{1}{\pi}\mathrm{Im}[\langle\psi_0|\hat{O}^\dagger \frac{1}{\omega + E_0 + i\epsilon - \hat{H}}\hat{O}|\psi_0\rangle]$$

Dagotto, Rev. Mod. Phys 66, 763 (1994)

Ex. Many-body excited spectrum calculation of $La_{3/2}Sr_{1/2}NiO$: multi-orbital extended Hubbard model, matrix size of M=64,000,000

S. Yamamoto, et al., JPSJ77, 114713 (2008)



Kω：solver of shifted Krylov-subspace method

(one of)target: excited spectrum calculation in quantum lattice model

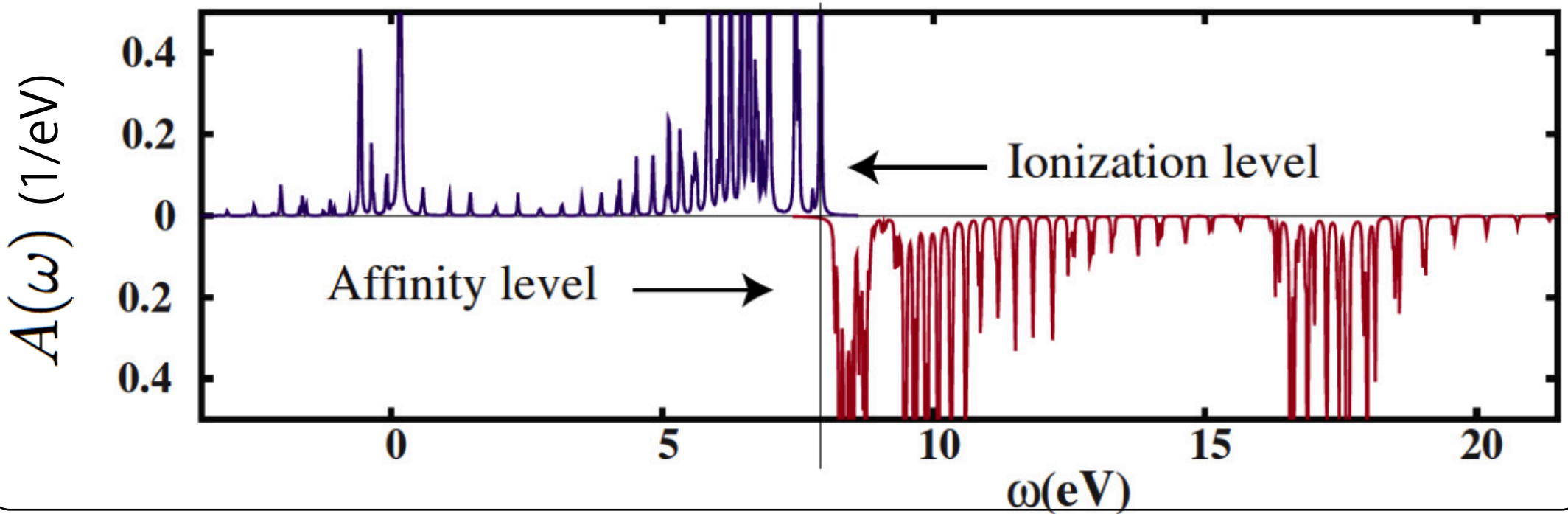
$$A(\omega) = -\frac{1}{\pi}\mathrm{Im}\left[\langle\psi_0|\hat{O}^\dagger \frac{1}{\omega + E_0 + i\epsilon - \hat{H}}\hat{O}|\psi_0\rangle\right]$$

shifted linear equations

$$(zI - H)\boldsymbol{x} = \boldsymbol{b}$$

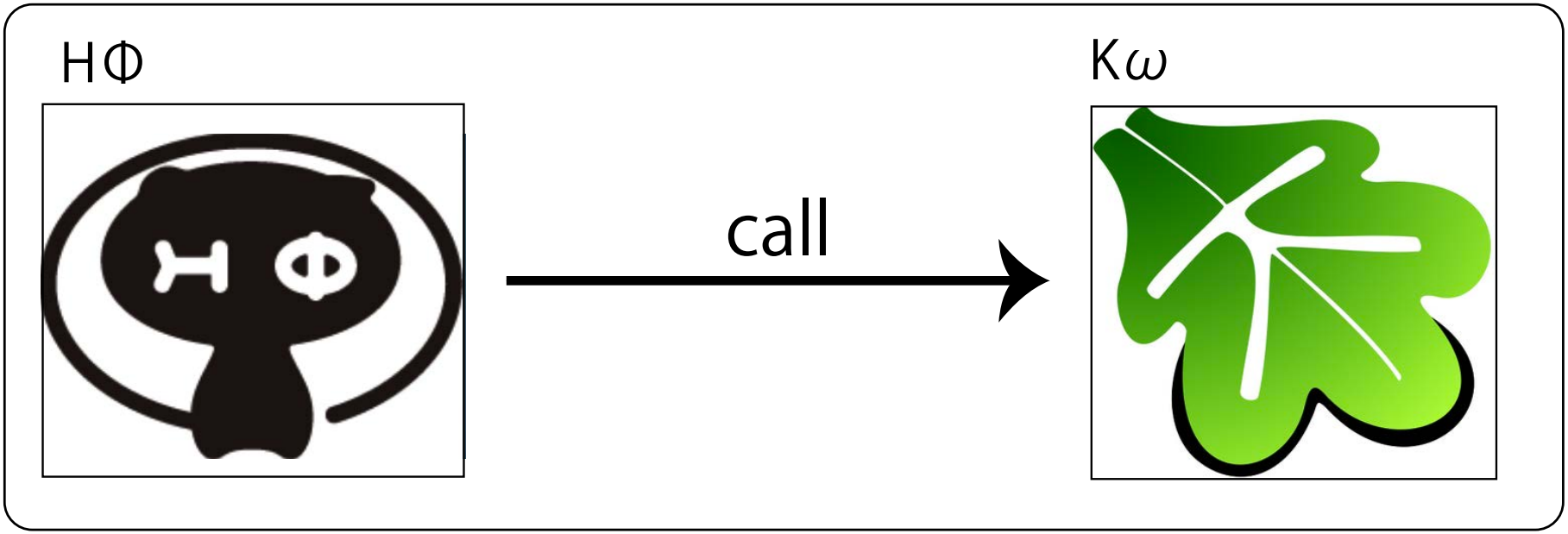
Ex. Many-body excited spectrum calc
multi-orbital extended Hubbard mo

S. Yamamoto, et al., JPSJ77, 114713 (2008)



Kω : solver of shifted Krylov-subspace method

- Code development with ISSP researchers in
  'Project for advancement of software usability in materials science' at 2016 [1,2]
- Primary target application: HΦ, quantum lattice model solver



[1] T. Hoshi, Y. Yamaji, M. Kawamura, K. Yoshimi, T. Misawa, S. Todo, T. Sogabe, and N. Kawashima, 'Joint Research Highlight: Development of Numerical Library Kω ver. 1 and Quantum Lattice Solver HΦ ver. 2' , Annual Report 2016 of ISSP, (2007)

[2] Kawamura et al., in preparation

Generality of shifted Krylov-subspace solvers in many physics areas

[1] **A. Frommer, Computing 70, 87 (2003)** (Theorem) → **QCD**

[2] R. Takayama, T. Hoshi, T. Sogabe, S.-L. Zhang, and T. Fujiwara, Phys. Rev. B 73, 165108 (2006). →**Large-scale eletronic state calculation (ELSES)**

[3] S. Yamamoto, T. Fujiwara, and Y. Hatsugai, Phys. Rev. B 76, 165114 (2007); S. Yamamoto, T. Sogabe, T. Hoshi, S.-L. Zhang and T. Fujiwara, J. Phys. Soc. Jpn., 77,114713 (2008).

→ **Many-body problem** [$La_{3/2}Sr_{1/2}NiO_4$] → **Kω (with HΦ)**

[4] S. Iwase, T. Hoshi, T. Ono, Phys. Rev. E 91, 063305, 9pp. (2015)

→ **Quantum transport calculation (RSPACE)** with Prof. Ono (Tsukuba U)

[5] F. Giustino, M. L. Cohen, S. G. Louie, PRB. 81, 115105 (2010)

→ **Ab initio excitation (GW) calculation**

[6] T. Mizusaki, K Kaneko, M. Honma, T. Sakurai, Phys. Rev.C 82, 024310 (2010)

→ **Shell model calculation**

[7] Y. Futamura, H. Tadano and T. Sakurai, JSIAM Letters 2, 127 (2010).

→ **Large-scale electronic state calculation with real-space grid (RSDFT)**

[8] Y. Nagai, Y. Shinohara, Y. Futamura, and T. Sakurai, JPSJ 86, 014708 (2017)

→ **Nano-structured superconducting system**

k-ep:
Computing the *k*-th eigenpair of large-scale generalized eigenvalue problems

T. Hoshi, . D. Lee (Nagoya U)

[1] D. Lee et al. J. J. Indust. Appl. Math. 30, 625-633 (2013)
[2] D. Lee et al. J. Comp. Phys. 371, 618-632 (2018)
https://doi.org/10.1016/j.jcp.2018.06.002
http://arxiv.org/abs/1710.05134

Code: https://github.com/lee-djl/k-ep

Outline

1. Introduction
   - The *k*-th eigenvalue problem
   - Existing eigenvalue problems & methods
2. Our approach
   - Sparse LDL factorization
   - Three-stage algorithm
3. Numerical experiments
   - Pentacene & other materials
4. Concluding remarks

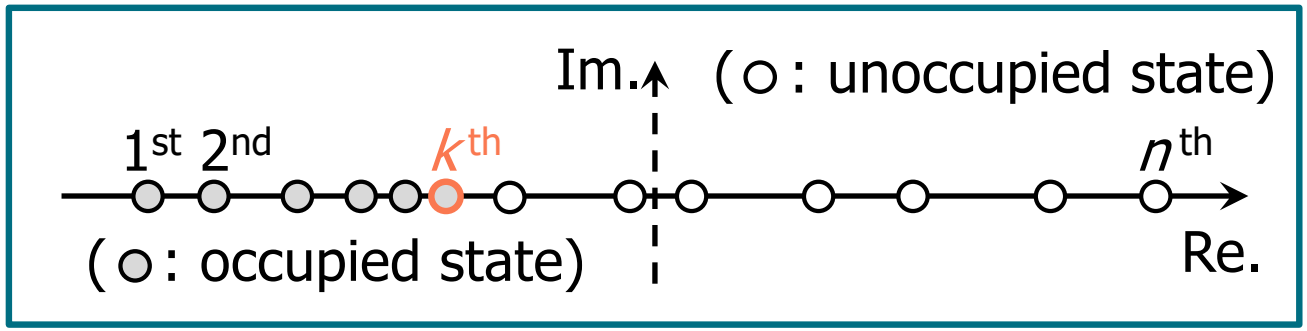
The *k*-th eigenvalue problem

Generalized Hermitian eigenvalue problem

For $n \times n$ large sparse Hermitian A, B and positive definite B,

and given target index k,

$$A\boldsymbol{x}_k = \lambda_k B\boldsymbol{x}_k.$$

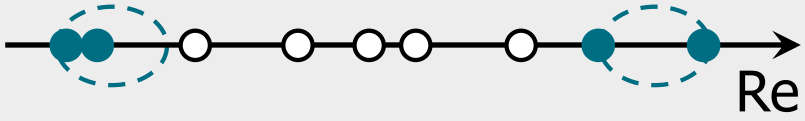
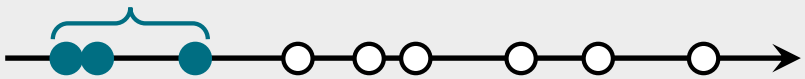
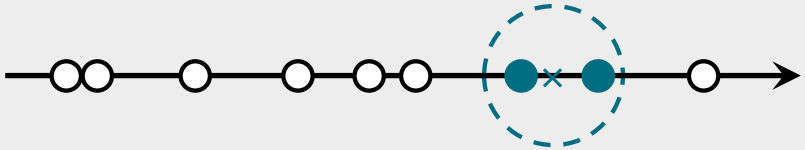


Target index k is not at either end of $[1,n]$.

Target index k → typically the highest-occupied level

Uniquely determined from N_{elec}, the number of electrons.

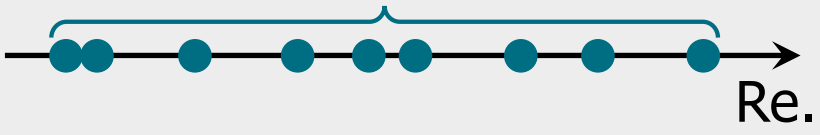
Existing eigenvalue problems & methods

A subset of eigenpairs		Subspace methods
Exterior		• Lanczos
Leftmost/ rightmost		• LOBPCG • Jacobi–Davidson
Near the target point		• Shift-and-invert • Jacobi–Davidson
Within the target interval		• Contour-integral (z-Pares, FEAST) • Filtering methods

Unable to compute the *k*-th eigenpair.

The *k*-th eigenpair is not the target of the subspace methods.

Existing eigenvalue problems & methods

The entire eigenpairs	Methods of dense transformations
	• ScaLAPACK • EigenExa • ELPA

1. Transform to a standard problem

   $B = CC^{\mathrm{H}}$, (e.g., Cholesky)

   $C^{-1}AC^{-\mathrm{H}} = A'$.

2. Solve the standard problem

   $QA'Q^{\mathrm{H}} = T$, (e.g., tridiagonalization)

   $T\boldsymbol{y} = \lambda\boldsymbol{y}$. (e.g., divide-and-conquer)

3. Back-transform eigenvectors

   $C^{-\mathrm{H}}Q^{\mathrm{H}}\boldsymbol{y} = \boldsymbol{x}$.

A practical size limit is $n \approx 10^6$.

A problem of $n = 10^6$ was solved on the full K computer in 1.5 hrs [Hoshi et al., 2016].

Outline

1. Introduction
   - The *k*-th eigenvalue problem
   - Existing eigenvalue problems & methods
2. *Our approach*
   - *Sparse LDL factorization*
   - *Three-stage algorithm*
3. Numerical experiments
   - Pentacene & other materials
4. Concluding remarks

Our approach

A subset of eigenpairs	The entire eigenpairs
Subspace methods	Methods of dense transformations
Unable to compute $(\lambda_k, \boldsymbol{x}_k)$	Practical size limit $(n \approx 10^6)$

Goal

Efficient computation of $(\lambda_k, \boldsymbol{x}_k)$ of large-scale $(n > 10^6)$ problems

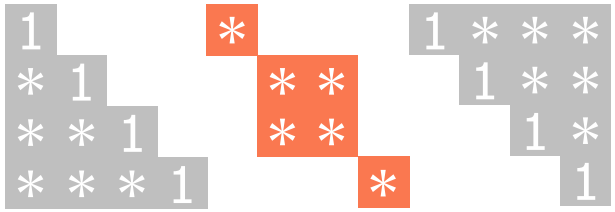
Idea

Use a subspace method & a sparse LDL factorization

Sparse LDL factorization

LDL factorization: Diagonalization by congruence transformation

$$QP\,A\,P^{\mathsf{T}}Q^{\mathsf{T}} = LDL^{\mathsf{H}} =$$



.

A : Hermitian,

P : permutation (fill-reducing),

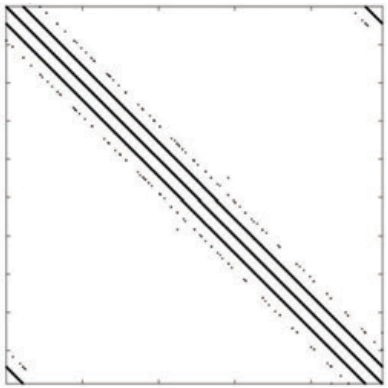
Q : permutation (pivot selection),

L : unit lower triangular,

D : block diagonal (block size 1 or 2).

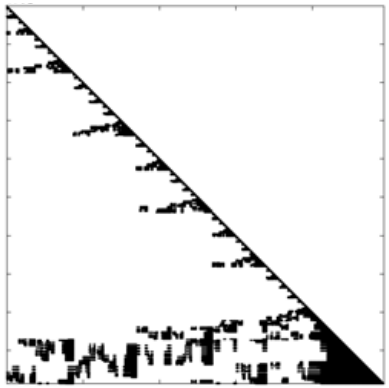
Case of PENTF98736 (968 pentacene molecules, n = 98736)

$A =$



Nonzero elements: 0.2%

$L =$

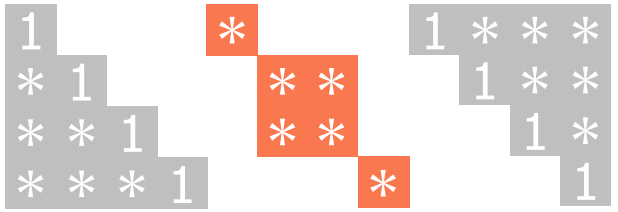


Nonzero elements: 6.2%

Sparse LDL factorization

LDL factorization: Diagonalization by congruence transformation

$$QP\,A\,P^{\mathsf{T}}Q^{\mathsf{T}} = LDL^{\mathsf{H}} =$$



.

A : Hermitian,

P : permutation (fill-reducing),

Q : permutation (pivot selection),

L : unit lower triangular,

D : block diagonal (block size 1 or 2).

Software libraries

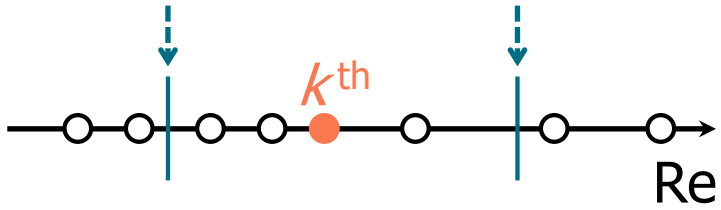
- PARDISO [https://www.pardiso-project.org/, Intel MKL built-in subroutine]
- MA57 [http://www.hsl.rl.ac.uk/, MATLAB built-in function]
- MUMPS [http://mumps.enseeiht.fr/]

Three-stage algorithm

Idea

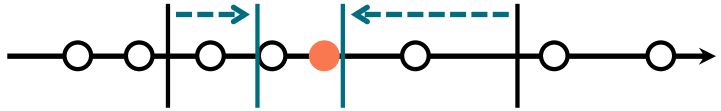
Use a subspace method & a sparse LDL factorization

Stage 1



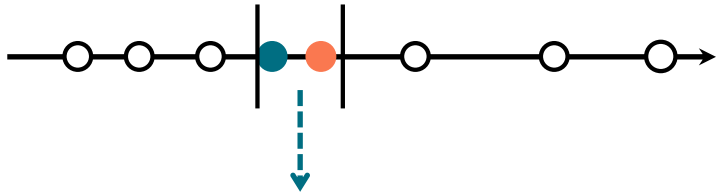
Find an interval containing λ_k

Stage 2



Narrow down the interval

Stage 3

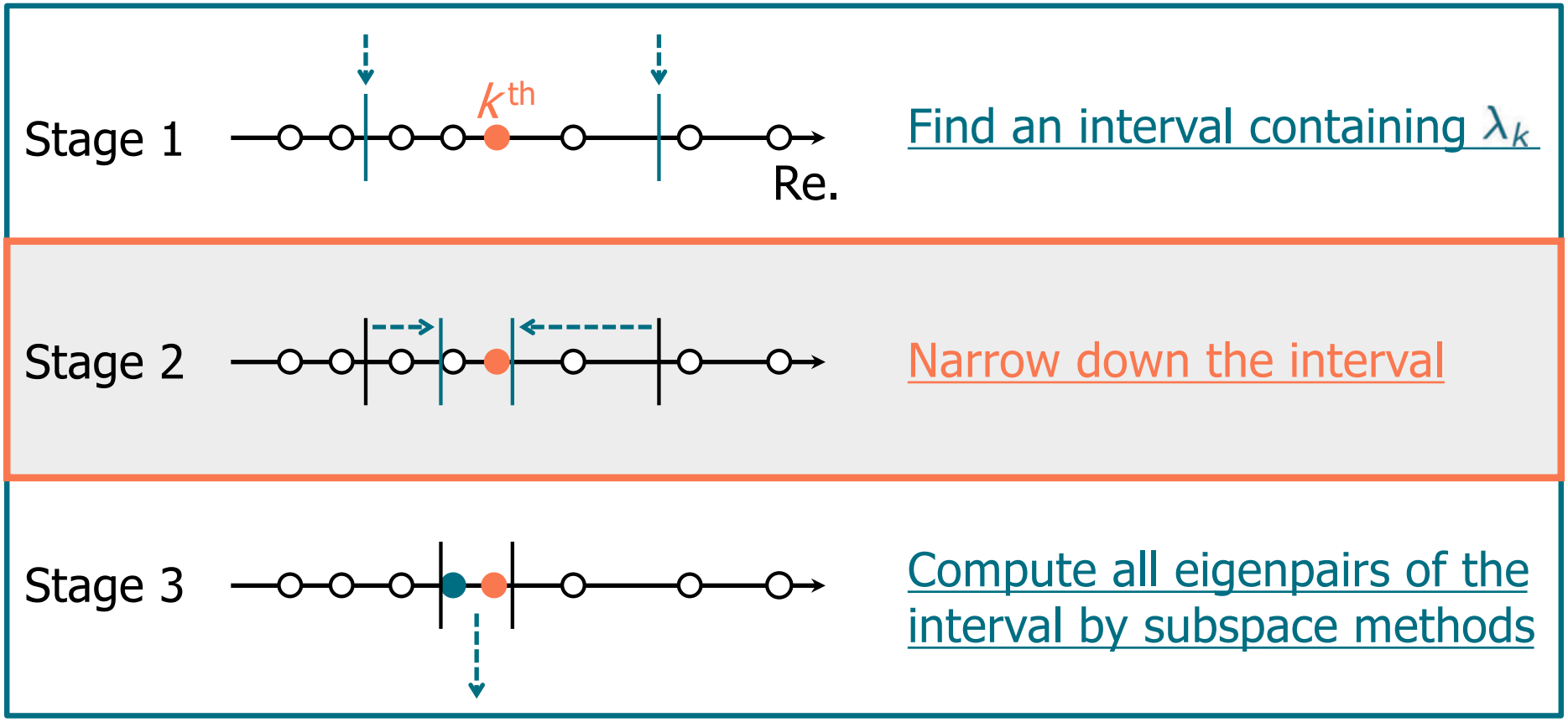


Compute all eigenpairs of the interval by subspace methods

Three-stage algorithm

Idea

Use a subspace method & a sparse LDL factorization

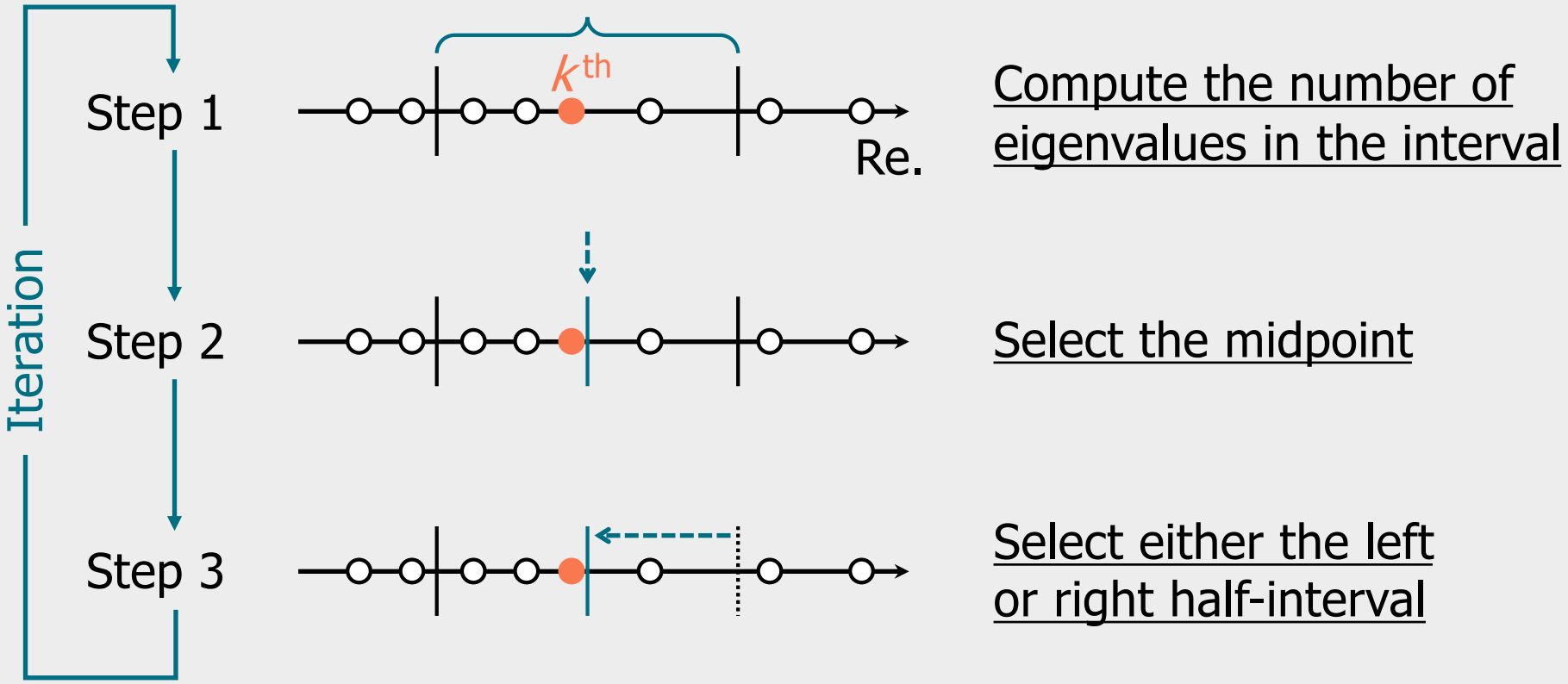


Use of sparse LDL factorization

Search λ_k using the result of factorization

Stage 2

Bisection: Narrow down the interval by half
until the number of eigenvalues in the interval becomes small



Use of sparse LDL factorization

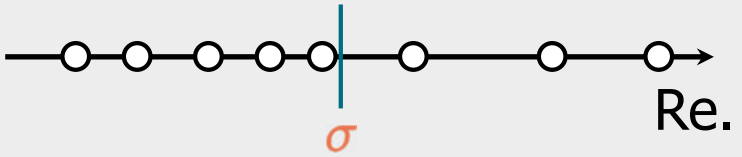
Count the number of eigenvalues smaller than given $\sigma \in \mathbb{R}$

Counting the eigenvalues

$n_\sigma(A, B)$: *Eigenvalue count :*

The number of eigenvalues smaller than given $\sigma \in \mathbb{R}$

Example
$(n = 8)$



$n_\sigma(A, B) = 5.$

Computing $n_\sigma(A, B)$

- Sparse LDL factorization (diagonalization) of shifted matrices

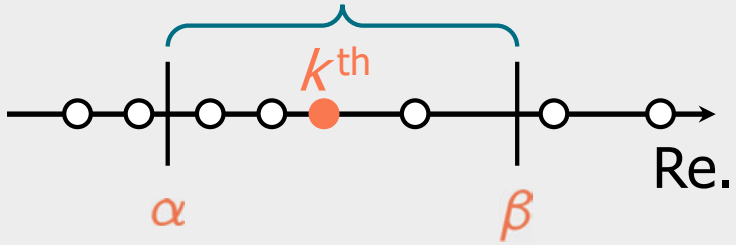
$$Q_\sigma P \left(A - \sigma B\right) P^{\mathsf{T}} {Q_\sigma}^{\mathsf{T}} = L_\sigma D_\sigma {L_\sigma}^{\mathsf{H}}.$$

- From Sylvester's theorem of inertia (1852),

$$n_\sigma(A, B) = n_0(D_\sigma, I)$$
$$= \text{The number of } D_\sigma\text{'s negative eigenvalues.}$$

Stage 2. Step 1 & 3

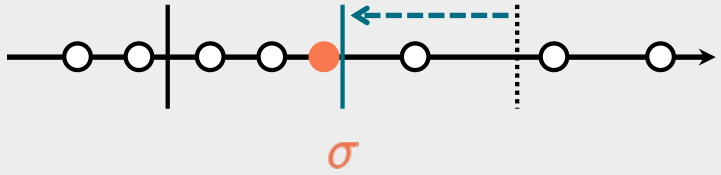
Step 1



Compute the number of eigenvalues in the interval

$$n_\beta(A, B) - n_\alpha(A, B) = n_0(D_\beta, I) - n_0(D_\alpha, I) \leq \text{Stopping criterion.}$$

Step 3

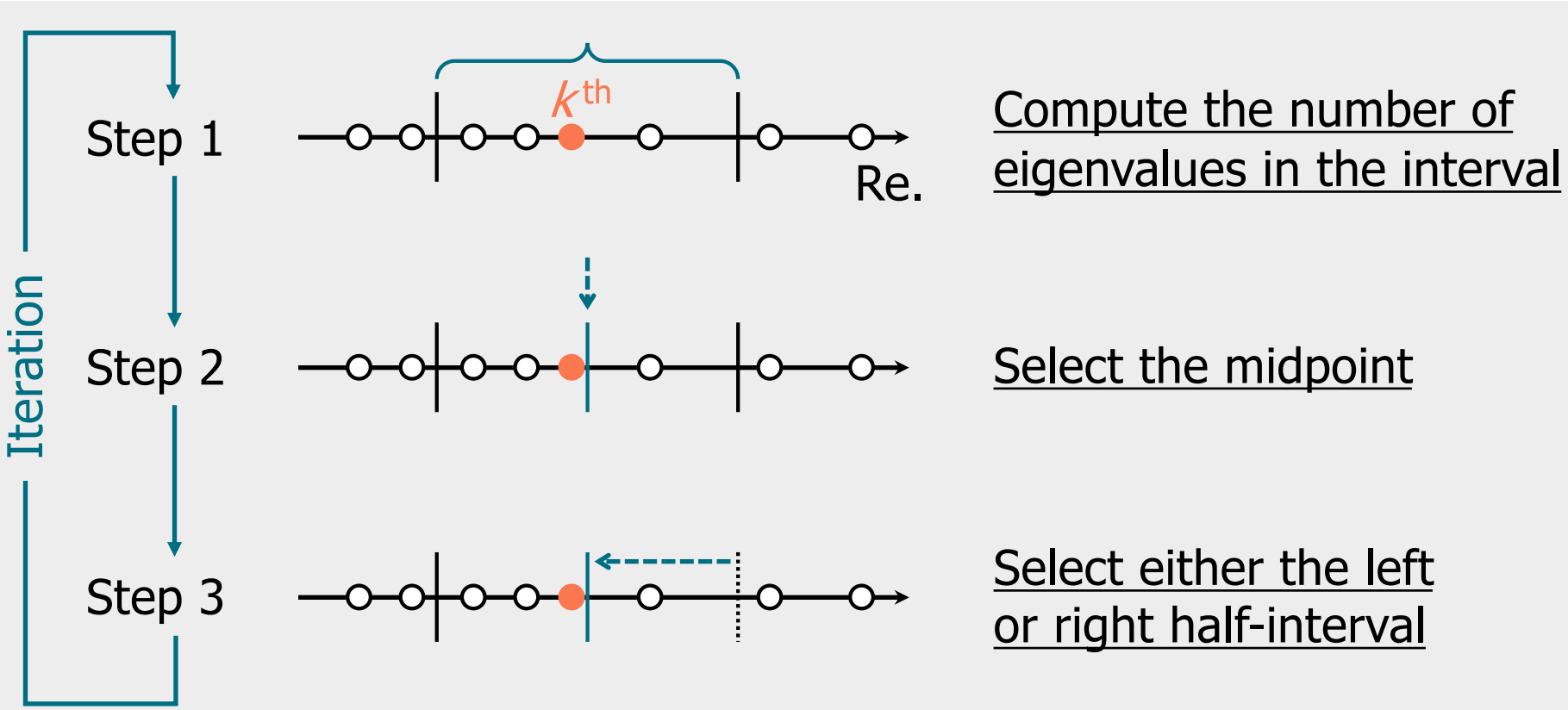


Select either the left or right half-interval

$$k \leq n_\sigma(A, B) \iff \lambda_k < \sigma.$$

Stage 2 (recap)

Bisection: Narrow down the interval by half
until the number of eigenvalues in the interval becomes small



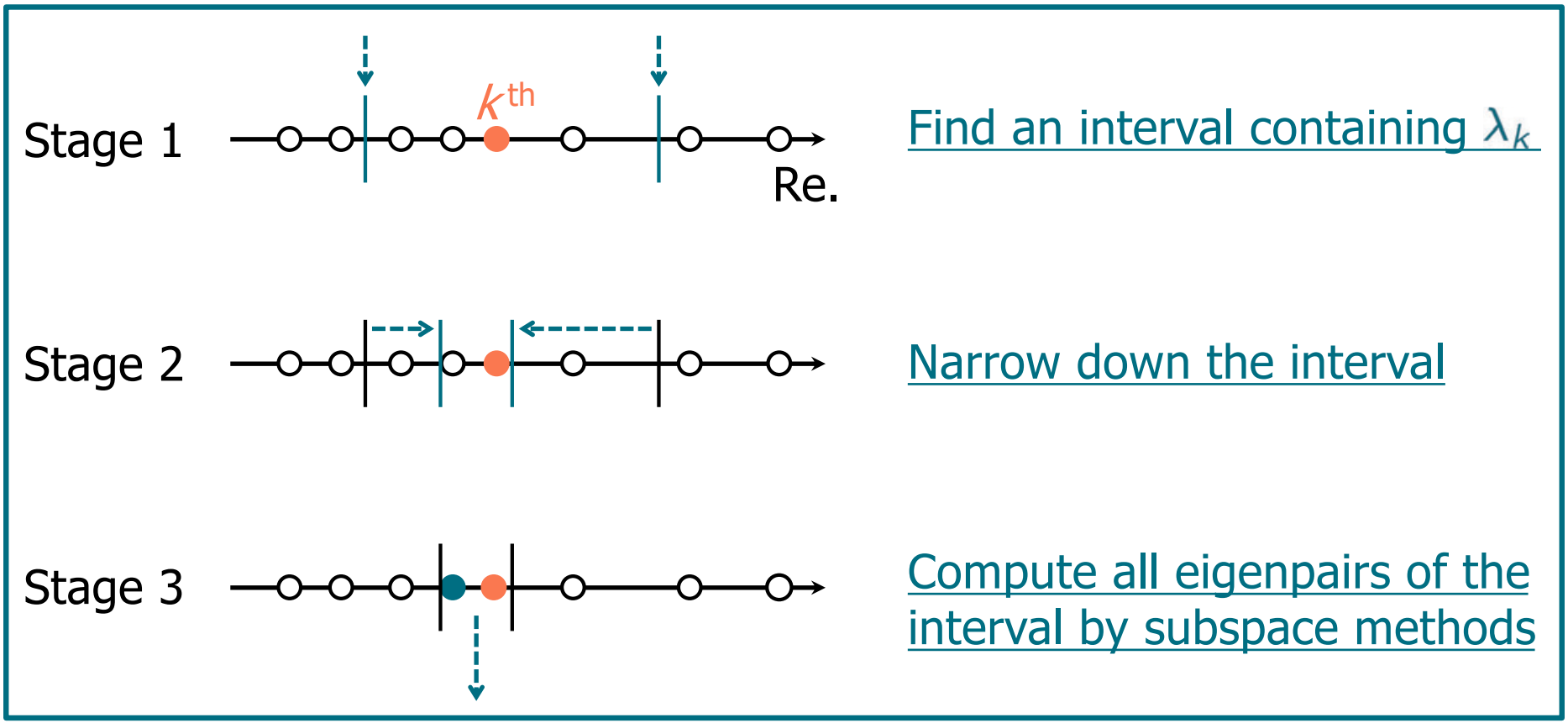
Use of sparse LDL factorization

Count the number of eigenvalues smaller than given $\sigma \in \mathbb{R}$

Three-stage algorithm (recap)

Idea

Use a subspace method & a sparse LDL factorization



Use of sparse LDL factorization

Search λ_k using the result of factorization

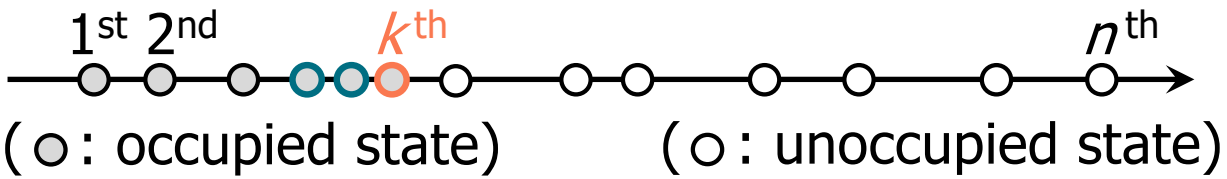
Outline

1. Introduction
   - The *k*-th eigenvalue problem
   - Existing eigenvalue problems & methods
2. Our approach
   - Sparse LDL factorization
   - Three-stage algorithm
3. *Numerical experiments*
   - *Pentacene & other materials*
4. Concluding remarks

Numerical experiments

Target eigenpairs

- The *k*-th eigenpair (highest occupied state) & 20 eigenpairs below it



Matrix data

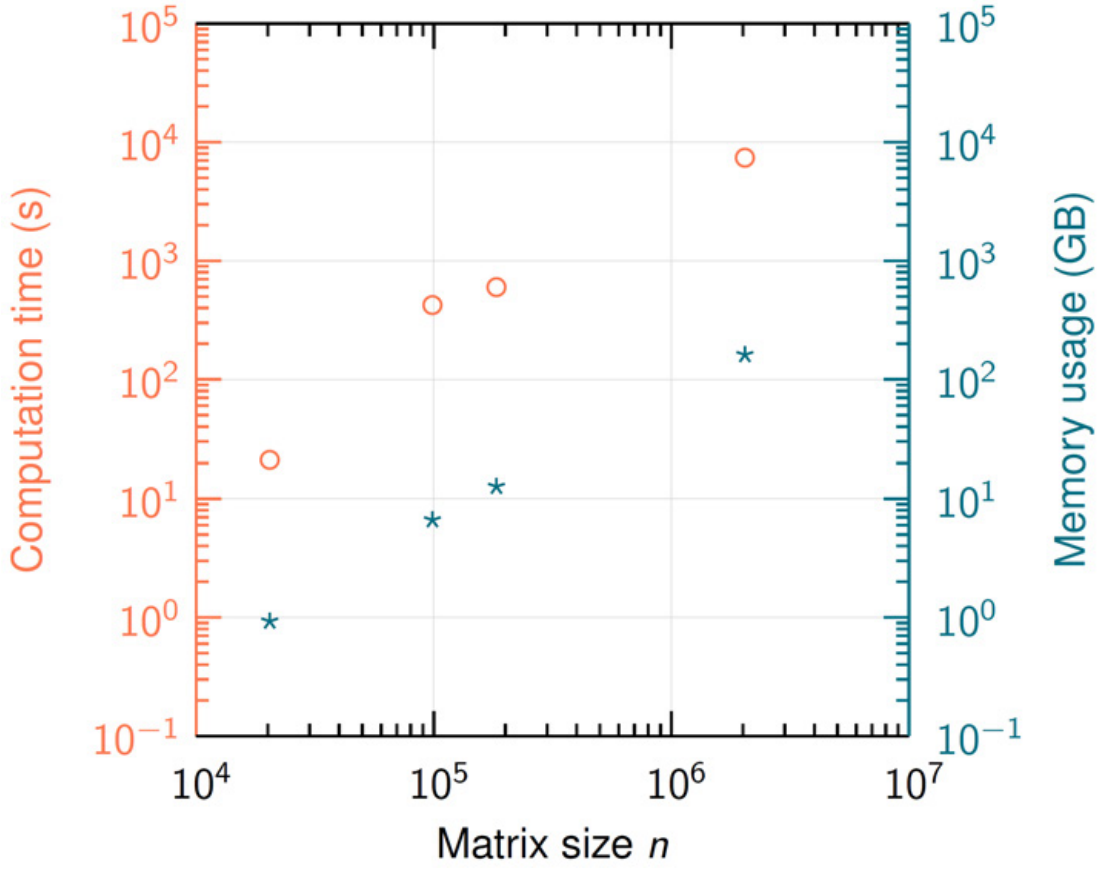
- Generated by ELSES quantum mechanical nanomaterial simulator
  [Hoshi et al., 2012; http://www.elses.jp/matrix/]

Computational environment & libraries

- ISSP System B Fat node (1 CPU, 10 cores, 10 OMP threads)
- Sparse LDL factorization: MUMPS (with METIS ordering)
- LAPACK, BLAS: Intel MKL (multi-threaded)

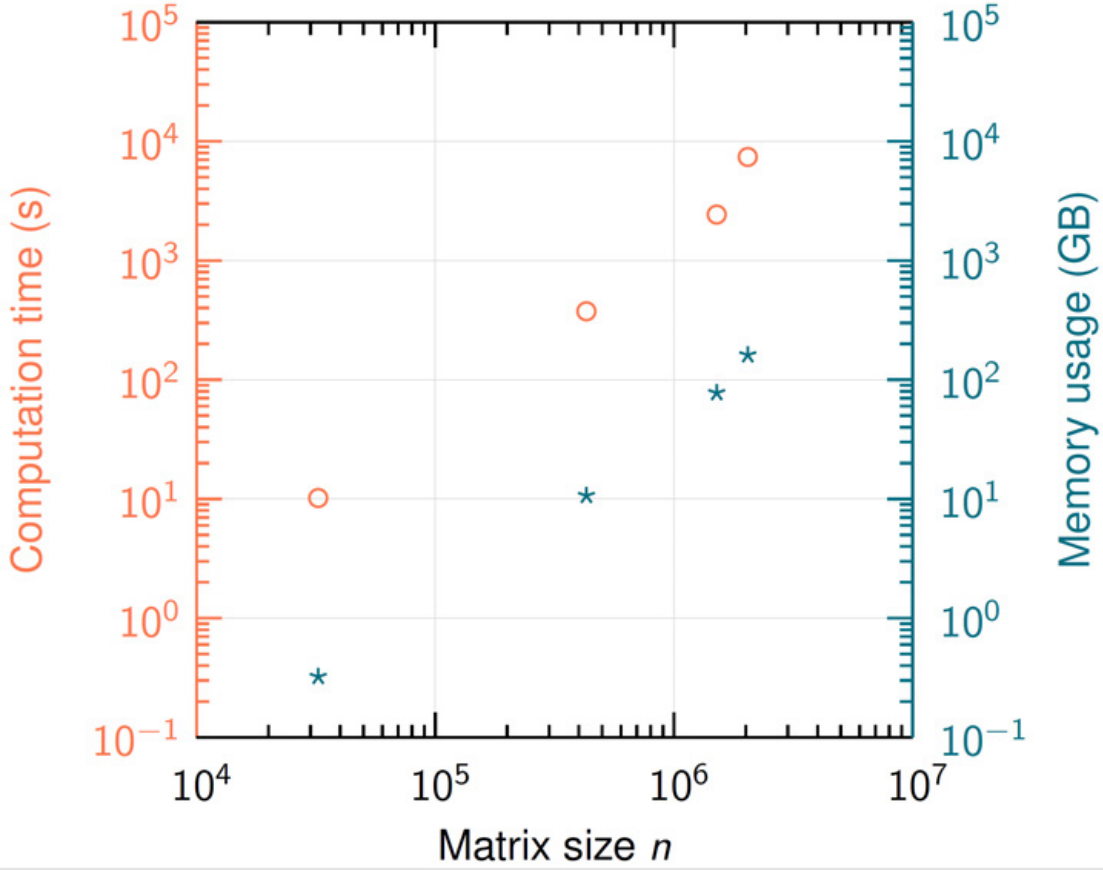
Pentacene thin film (PENTF)

#molecules	200	968	1,800	20,000
Size n	20,400	98,736	183,600	2,040,000
Target k	10,200	49,368	91,800	1,020,000
#nonzeros	2 M	9 M	16 M	182 M



PENTF & other materials

Material	Poly(Phenylene-Ethynylene)	Nano-composite carbon solid	Vibrating carbon nanotube	Pentacene thin film
Size n	32,346	430,080	1,512,000	2,040,000
Target k	16,173	215,040	336,000	1,020,000
#nonzeros	0.9 M	11 M	295 M	182 M



Detailed computational costs (1/2)

D. Lee et al, J. Comp. Phys. 371, 618-632 (2018)

Table 3
Computation time and computational resources consumed by Algorithm 4, its variant, and dense eigensolvers.

Data	Time (s)			#Core			Memory (MB)		
	Alg. 4	Ger.	Dense	Alg. 4	Ger.	Dense	Alg. 4	Ger.	Dense
APF4686	0.3	0.8	82.1	$1^{(w)}$	$1^{(w)}$	$1^{(w)}$	13	12	629
AUNW9180	19.4	69.0	655.2			$1^{(w)}$	267	245	2836
CPPE32346	4.4	194.0	1366.8			$32^{(K)}$	121	116	*33480*
NCCS430080	2024	109597	10586			$180000^{(K)}$	5069	5055	*5919002*
VCNT1512000	5132	602525	n/a			n/a	38242	31618	n/a

(w) workstation with Xeon E5-2690 (2.90 GHz).
(K) K computer with SPARC64 VIIIfx (2.00 GHz) and Tofu interconnect.

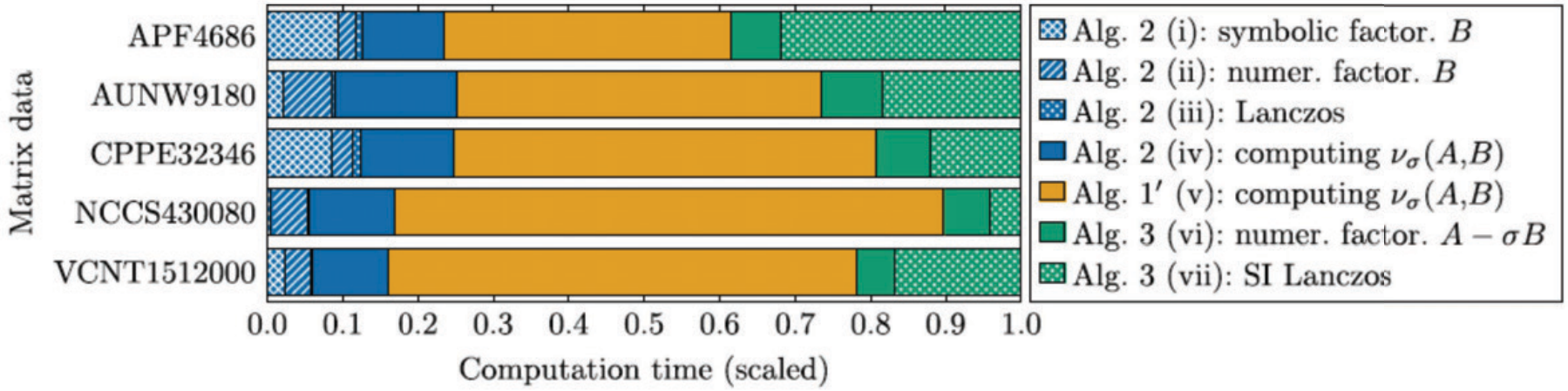


Fig. 4. Computation time of algorithms and computational tasks.

Detailed computational costs (2/2)

Elapsed times appear on an output file.
Ex. The case with the matrix 'APF4686’

```
 -----Computation time (s)
Overall                         :      0.4300000

Step 1 Task (i)   symbolic factor. B  :      0.0291000
Step 1 Task (ii)  numer. factor. B    :      0.1089000
Step 1 Task (iii) Lanczos             :      0.0093000
Step 1 Task (iv)  inertia computation :      0.0261000
Step 2 Task (v)   inertia computation :      0.0614000
Step 3 Task (vi)  numer. factor. A-sB :      0.0127000
Step 3 Task (vii) SI Lanczos          :      0.1825000

 -----Iteration count
Step 1 :   2
Step 2 :   5
Step 3 :  64
```

Concluding remarks

The k-th eigenvalue problem

Compute the *k*-th eigenpair

(e.g., the highest occupied energy level and its wave function)

of large sparse generalized Hermitian eigenvalue problems

Future versions

v.2 (i) multiple eigenpairs

(ii) skip of the first stage with a given initial interval

(iii) MPI-based LDLT factorization (MUMPS)

v.3 hierarchal parallelism

outer loop : for each set of the target levels (ideal parallelism)

inner loop : MPI-based routines

for LDLT factorization and other procedures