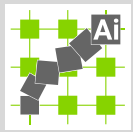
Recent algorithmic developments in ELPA

Bruno Lang



MATHEMATICAL MODELLING,
ANALYSIS AND
COMPUTATIONAL MATHEMATICS

BERGISCHE
UNIVERSITÄT
WUPPERTAL

Outline

The project

Eigenvalue SoLvers for Petaflop Applications – Algorithmic Extensions and Optimizations

ELPA-AEO

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

through IKT 2020 – Höchstleistungsrechnen

MPCDF Max Planck Computing & Data Facility

P. Kůs, **H. Lederer**, **A. Marek**

Bergische Universität Wuppertal

M. Galgon, **B. Lang**, V. Manin

H.-J. Bungartz, **Th. Huckle**, M. Rippl

Fritz-Haber-Institut der Max-Planck-Gesellschaft

Ch. Carbogno, **M. Scheffler**, D. Simoes Brambila

S. Köcher, **K. Reuter**, **Ch. Scheurer**

The ELPA library I

- Created during the ELPA project 

(12/2008–11/2011)

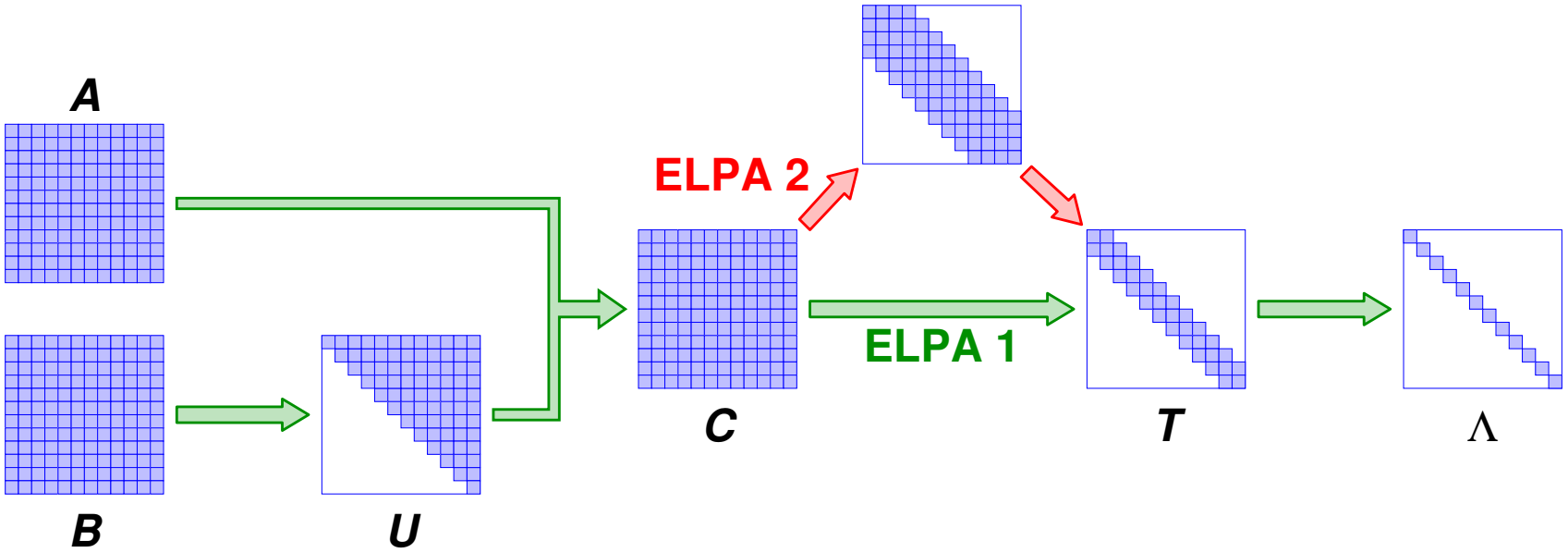
- Addresses the generalized hpd/spd eigenproblem

$$Ax = Bx\lambda$$

and the standard procedure for its solution

  - Cholesky decompositon $B =: U^H U$
    (or, e.g., "backward" $L^H L$; then $U \leftrightarrow L$ in the following)
  - $C := U^{-H} A U^{-1}$
    (then GEP $Ax = Bx\lambda \quad \sim \quad$ SEP $Cy = y\lambda$, where $y = Ux$)
  - Reduce C to tridiagonal form: $Q^H C Q =: T$ with unitary Q
  - Solve tridiagonal eigenproblem $Tz = z\lambda$
  - Back-transform eigenvectors to C: $y := Qz$
  - Back-transform eigenvectors to (A, B): $x := U^{-1} y$

The ELPA library II



and (back transforms of) evecs

- Key application: (eigenprobs mainly from SCF cycles)
- MPI + partial support for multi-threading
- Maintained and improved since then

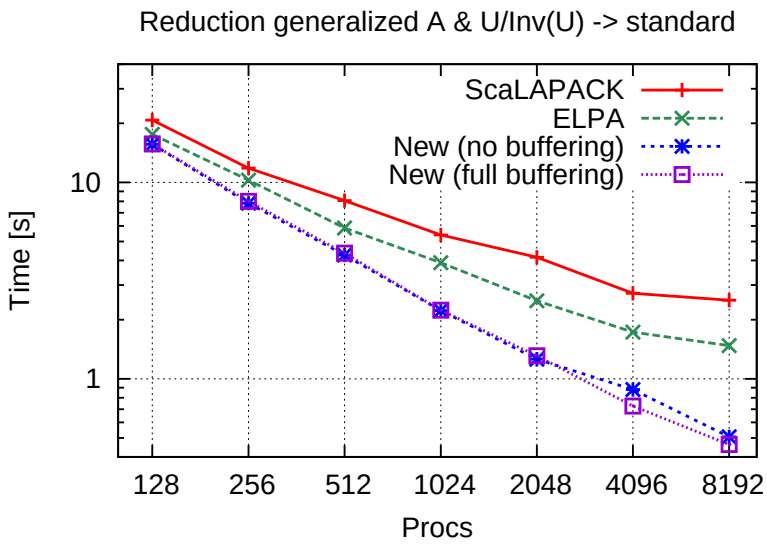
Outline

Motivation

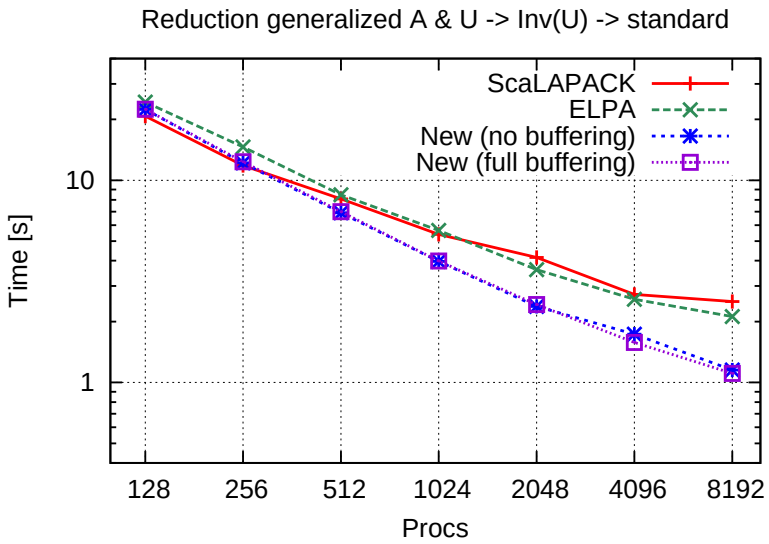
- Reduction generalized $\Rightarrow$ standard: $C := U^{-H} A U^{-1}$, where $B =: U^H U$
- Corresponding back-transform

Motivation

- Reduction generalized $\Rightarrow$ standard: $C := U^{-H} A U^{-1}$, where $B =: U^H U$
- Corresponding back-transform



"multiple GEPs, same B"
($n = 30{,}000$)



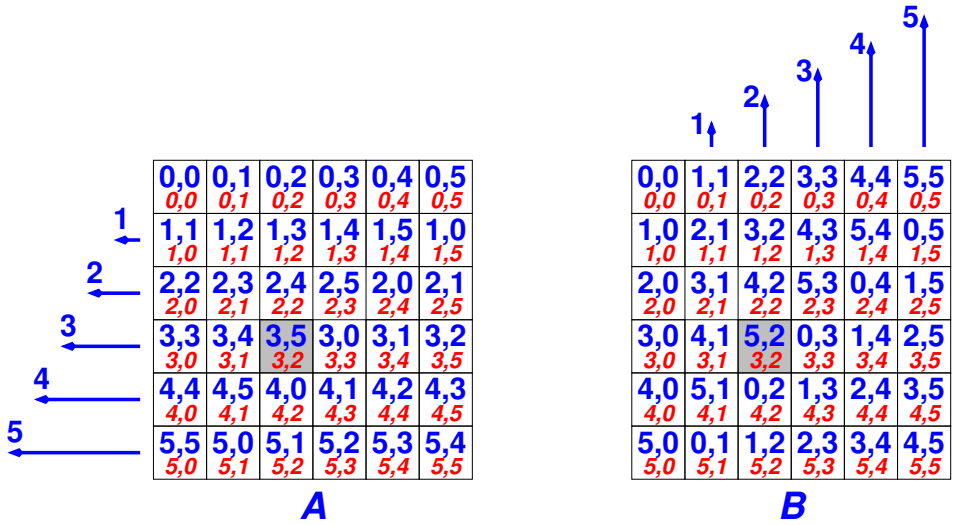
"single GEP"
(not main target of ELPA)

- **Matrix multiplications** with a Cannon-based approach

Cannon's algorithm (textbook case)

L.E. Cannon, Ph.D. thesis, Montana State University, Bozeman, MT (1969)

- **Initial skewing** of the blocks of A and B:



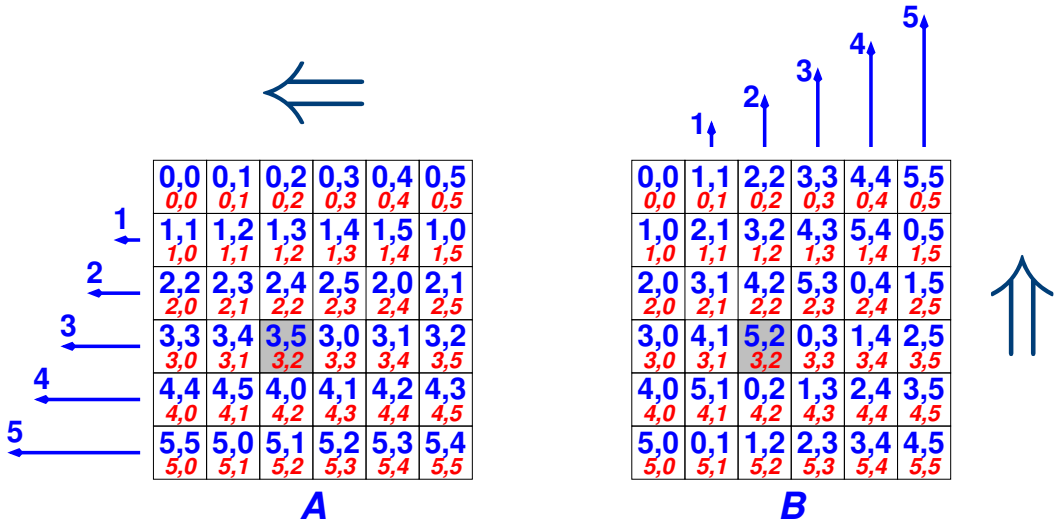
(block number, *process number*)

$$C_{3,2} = \sum_{k=0}^{5} A_{3,k} B_{k,2}$$
$$= A_{3,5} B_{5,2} +$$

Cannon's algorithm (textbook case)

L.E. Cannon, Ph.D. thesis, Montana State University, Bozeman, MT (1969)

- **Initial skewing** of the blocks of A and B:



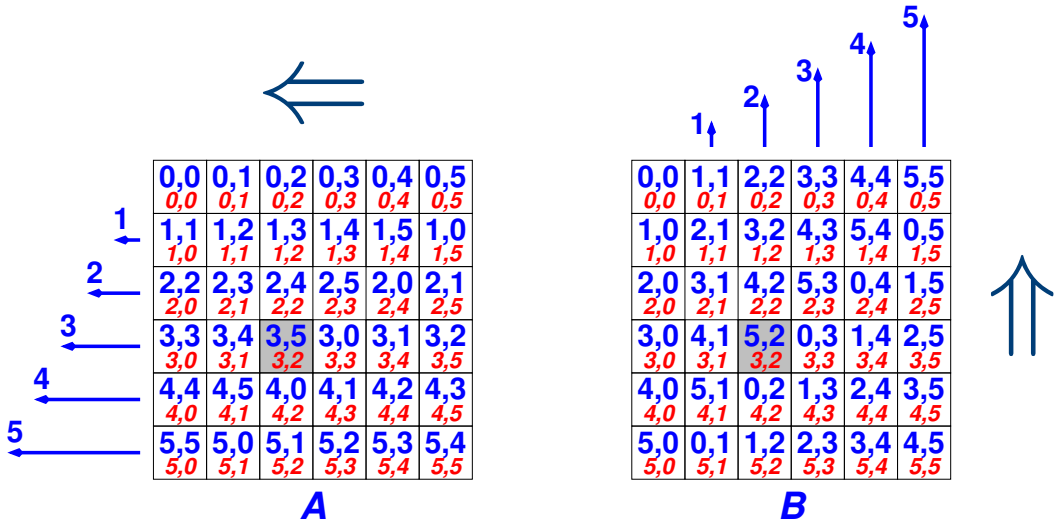
(block number, *process number*)

$$
\begin{aligned}
C_{3,2} &= \sum_{k=0}^{5} A_{3,k} B_{k,2} \\
&= A_{3,5} B_{5,2} + A_{3,0} B_{0,2} + \\
\end{aligned}
$$

Cannon's algorithm (textbook case)

L.E. Cannon, Ph.D. thesis, Montana State University, Bozeman, MT (1969)

- **Initial skewing** of the blocks of A and B:



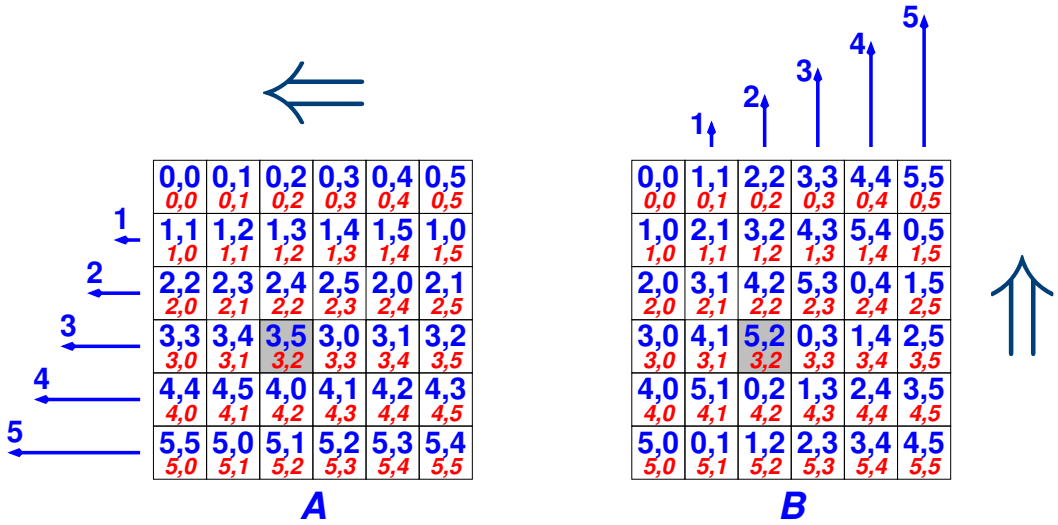
(block number, *process number*)

$$
\begin{aligned}
C_{3,2} &= \sum_{k=0}^{5} A_{3,k} B_{k,2} \\
&= A_{3,5}B_{5,2} + A_{3,0}B_{0,2} + A_{3,1}B_{1,2} +
\end{aligned}
$$

Cannon's algorithm (textbook case)

L.E. Cannon, Ph.D. thesis, Montana State University, Bozeman, MT (1969)

- **Initial skewing** of the blocks of A and B:

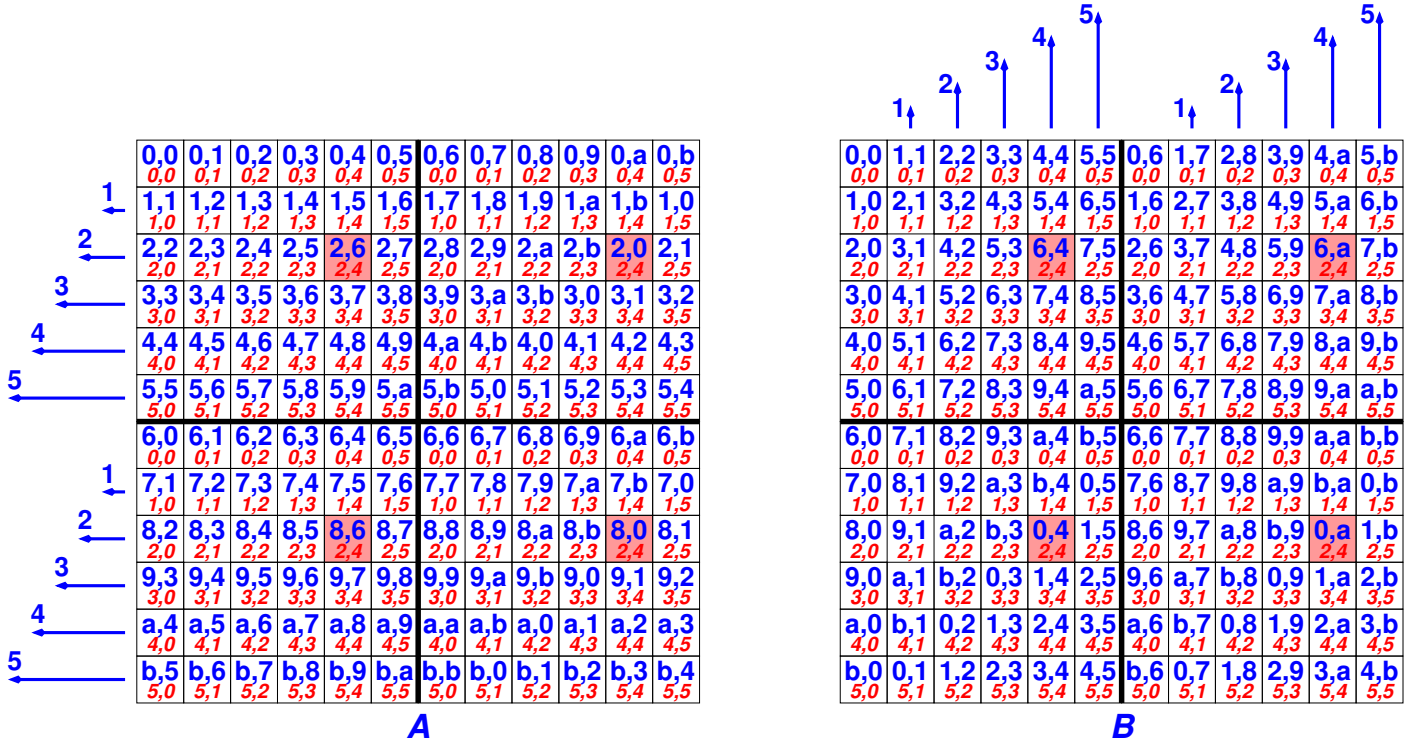


(block number, *process number*)

$$C_{3,2} = \sum_{k=0}^{5} A_{3,k} B_{k,2}$$
$$= A_{3,5}B_{5,2} + A_{3,0}B_{0,2} + A_{3,1}B_{1,2} + \ldots + A_{3,4}B_{4,2}$$

Cannon's algorithm (cont'd)

- **Shifts** instead of collective communication
- Extends easily to block cyclic distribution (ScaLAPACK, ELPA) on square grids



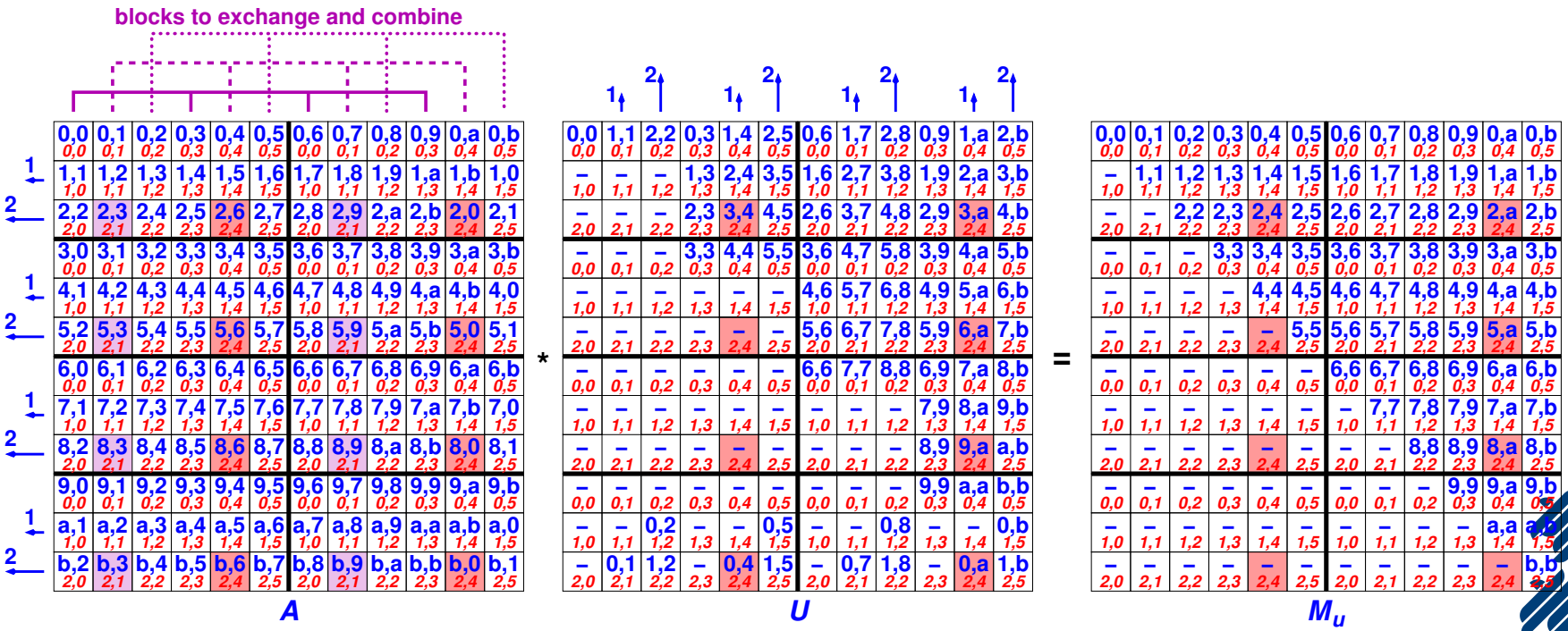
- Has been extended to non-square grids

E.g., H.-J. Lee, J.P. Robertson, J.A.B. Fortes, Proc. ICS '97, pp. 44–51

Multiplication 1: (upper triangle M_u of) $M = A \cdot U$ I

[V. Manin, BL, 2018 / MPCDF]

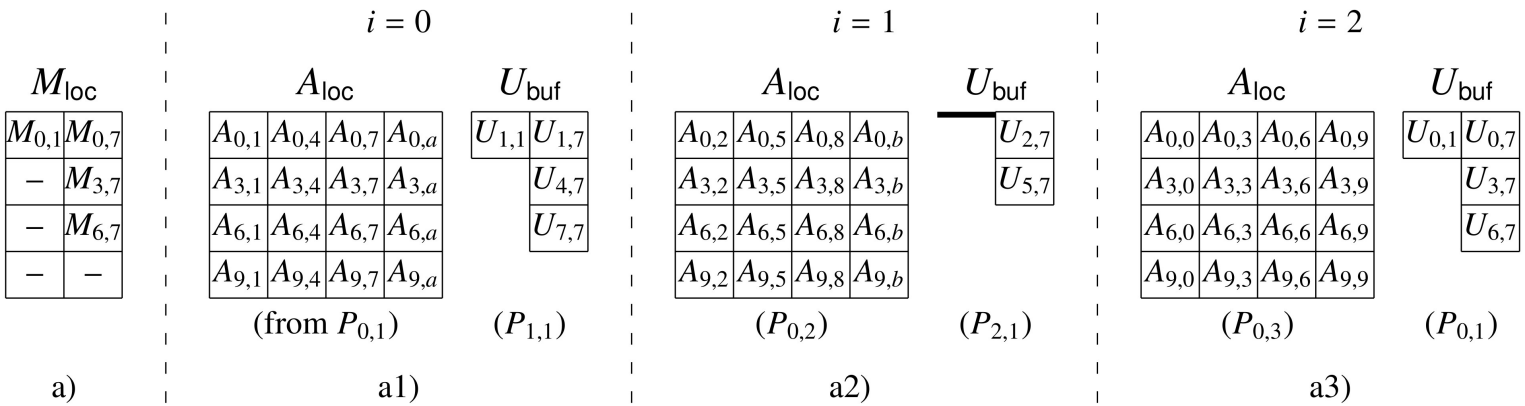
"Global view:"



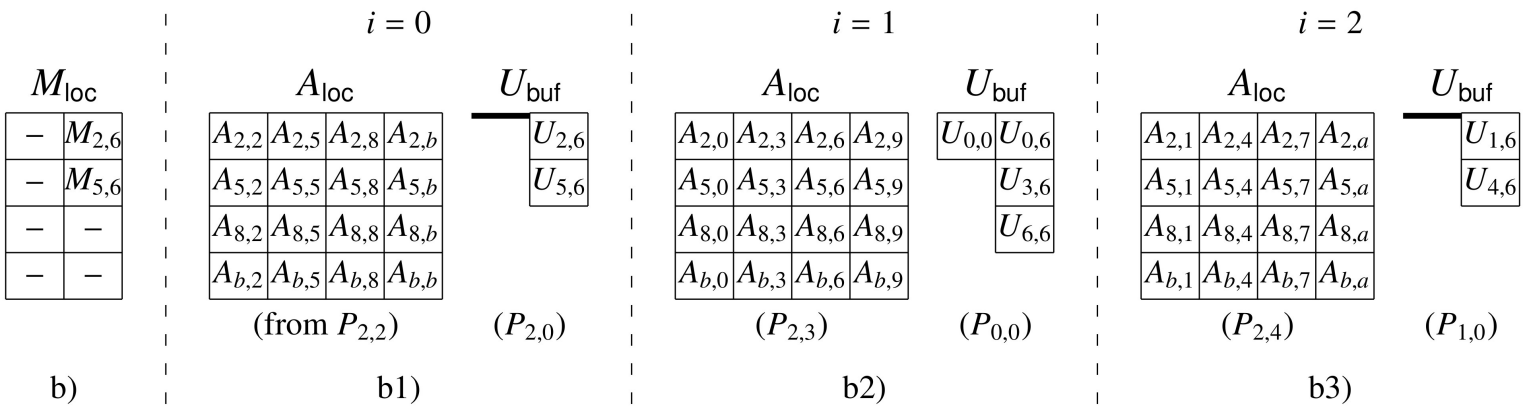
Multiplication 1: (upper triangle M_u of) $M = A \cdot U$ II

"Local view:"

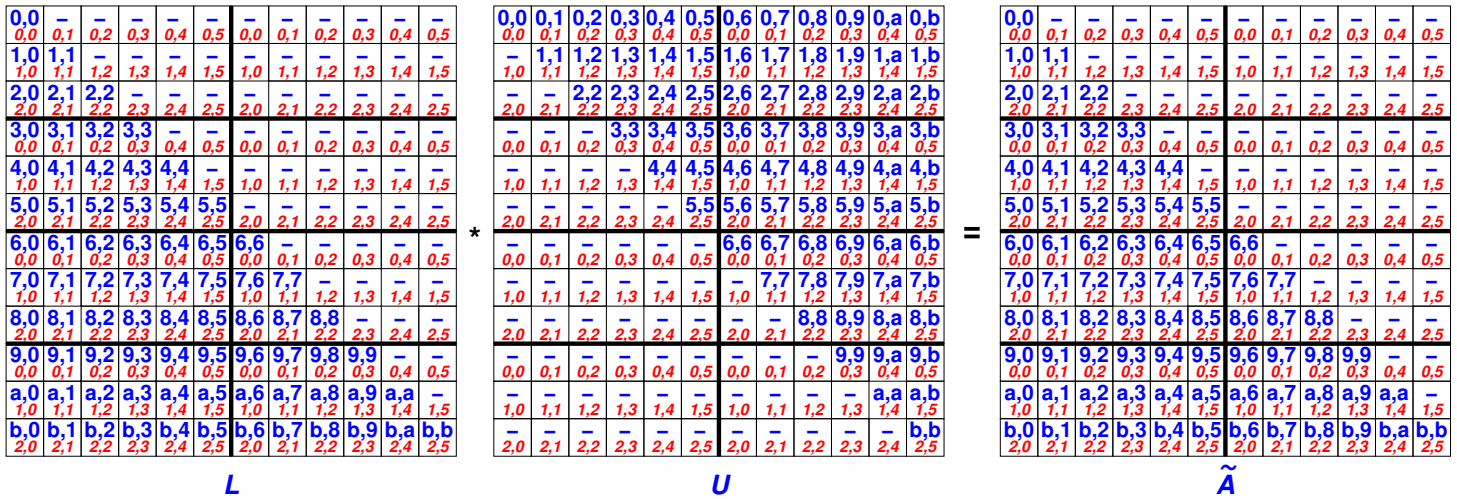
Situation in $P_{0,1}$:

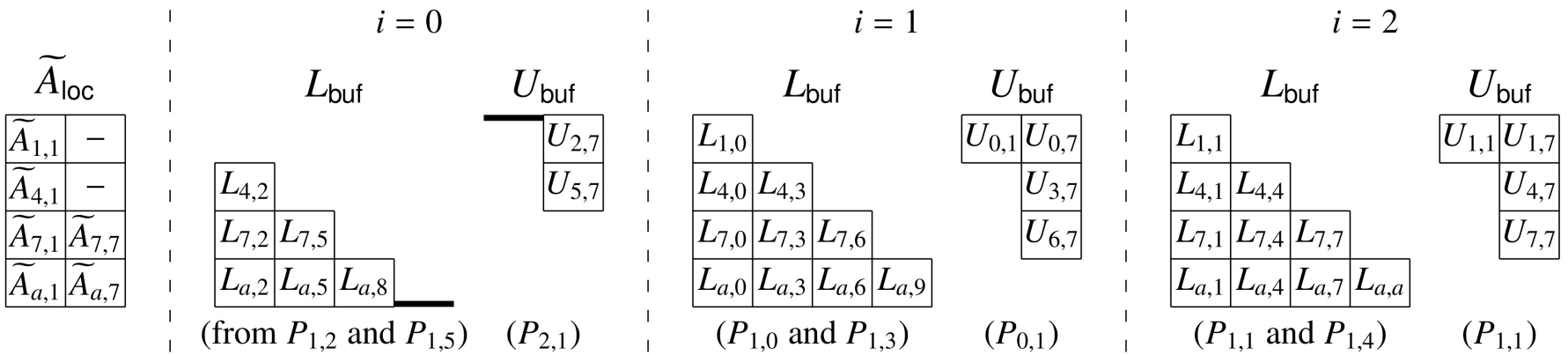


Situation in $P_{2,0}$:



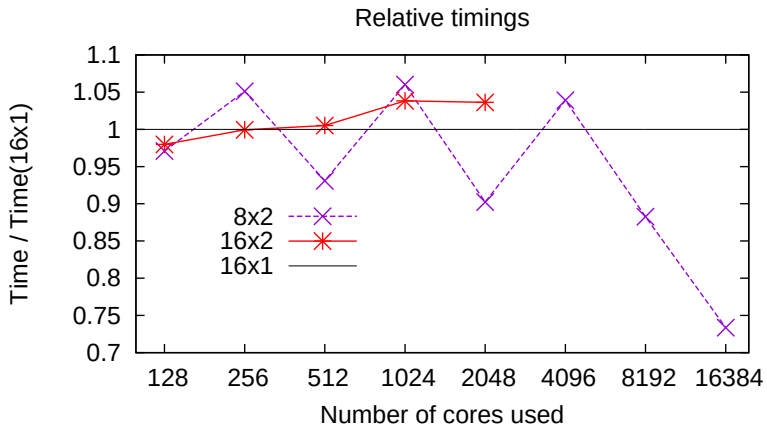
Multiplication 2: (lower triangle of) $\widetilde{A} = M_u^* \cdot U$





Complete transformation

- Optimized for $p_r \times p_c$ process grids with integer aspect ratio p_c/p_r (e.g., 3×3, 3×6, but not 3×4, 6×3)
- Combining both multiplications (and the transposes) allows for additional savings in communication
- Back-transformation of eigenvectors: similar
- "Generalized eigenproblem driver" will automatically select best method
- Potential for mixed "MPI + threads" parallelism



Outline

Motivation I

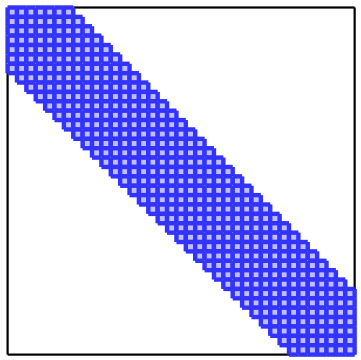
$$Ax = Bx\lambda, \qquad A \text{ Herm., } B \text{ Herm. pos. def.}$$

The standard procedure for the "full" case

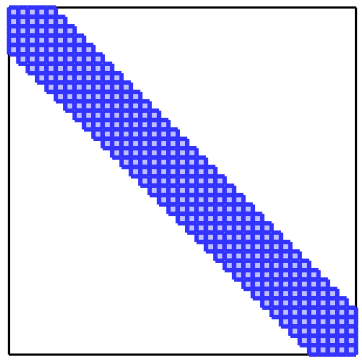
1. ("Reverse") Cholesky decomposition $B =: L^H L$
2. $C := L^{-H} A\, L^{-1}$
3. Reduce C to tridiagonal form: $Q^H C Q =: T$ with unitary Q
4. Solve tridiagonal eigenproblem $Tz = z\lambda$
5. Back-transform eigenvectors to C: $y := Qz$
6. Back-transform eigenvectors to (A, B): $x := L^{-1} y$

in principle also applies if A and B are banded, ...

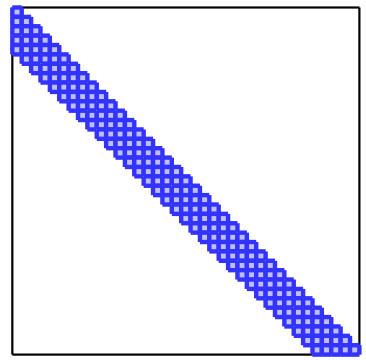
Motivation I



A



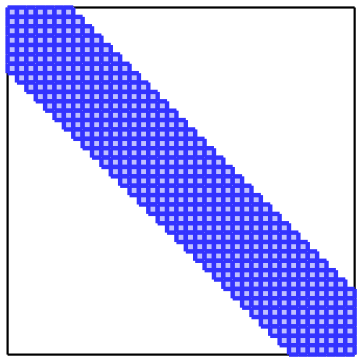
B



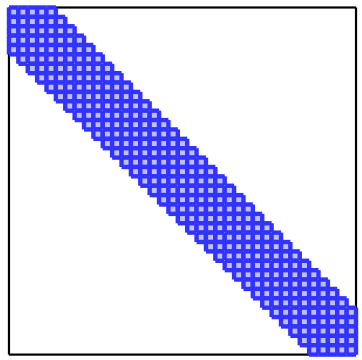
L

But:

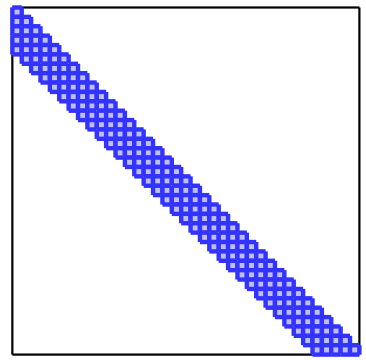
Motivation I



A

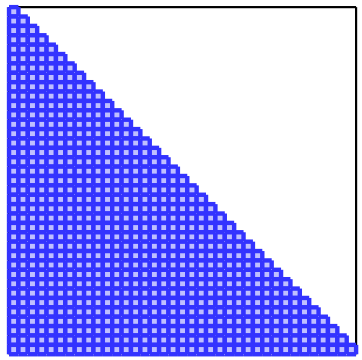


B



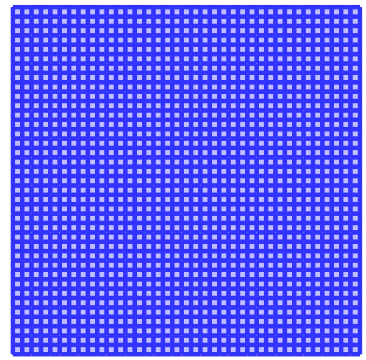
L

But:



L^{-1}

$\Rightarrow$



C

⊖ Standard eigenproblem $Cy = y\lambda$ is full

Band-preserving reduction I

[BL, 2018]

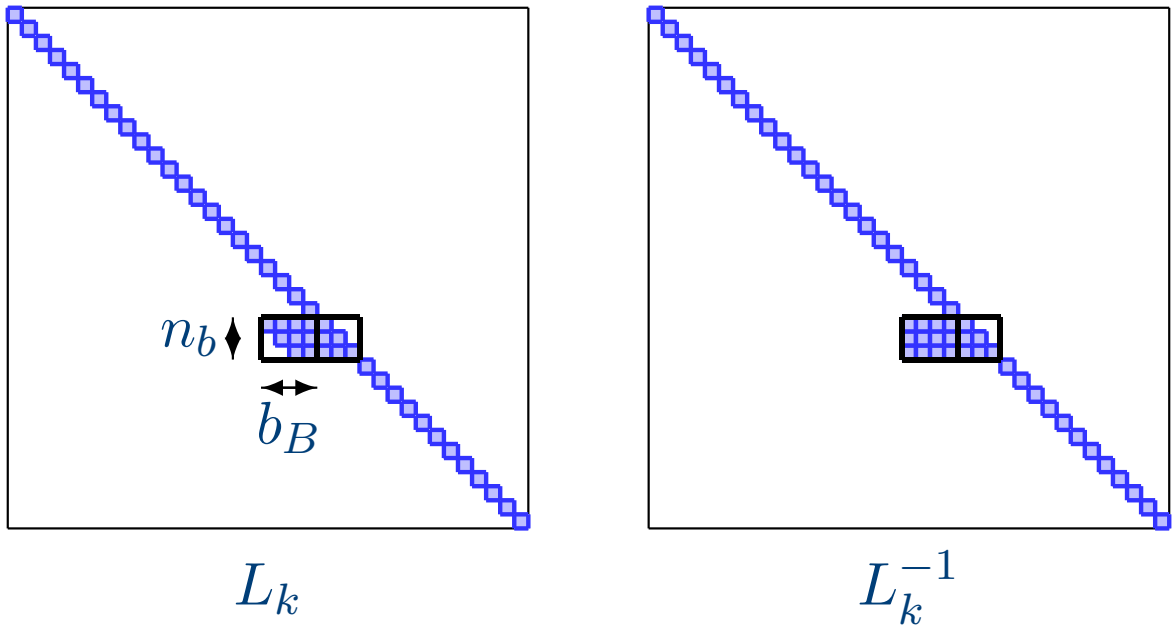
- Combines and extends ideas from
  - C.R. Crawford: *Reduction of a band-symmetric generalized eigenvalue problem.* Comm. ACM **16**(1):41–44, 1973.
  - LAPACK, version ≥ 3
- Blocked computations
- More flexibility ($b_B \leq b_A$, n_b, ...)
  (in the following pics: $b_A = 6, b_B = 4$, $n_b = 3$)
- Twisted factorizations

Band-preserving reduction II

Observation:

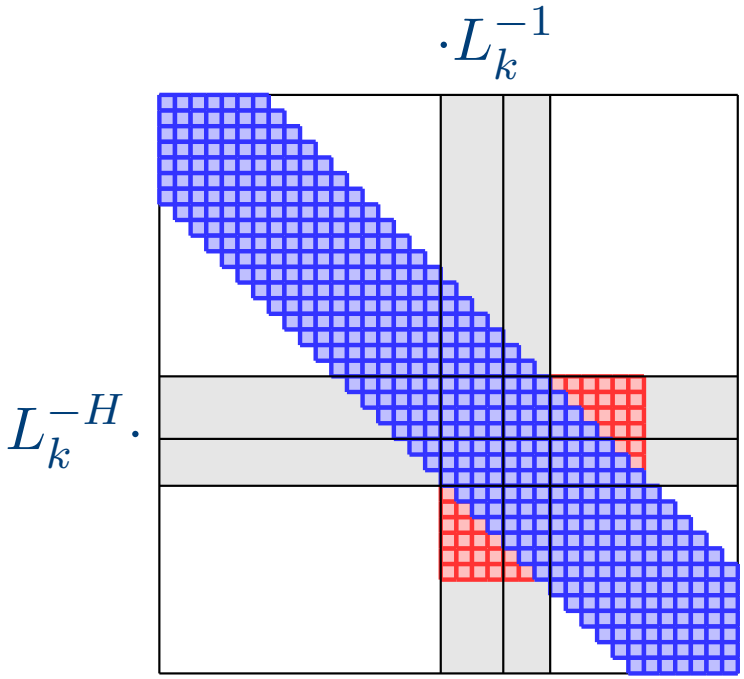
$$L = L_1 \cdots L_{k_{\max}} \qquad \text{and thus} \qquad L^{-1} = L_{k_{\max}}^{-1} \cdots L_1^{-1},$$

where



$\Rightarrow$ apply the L_k^{-1} to A (in the order $k_{\max}, \ldots, 1$)

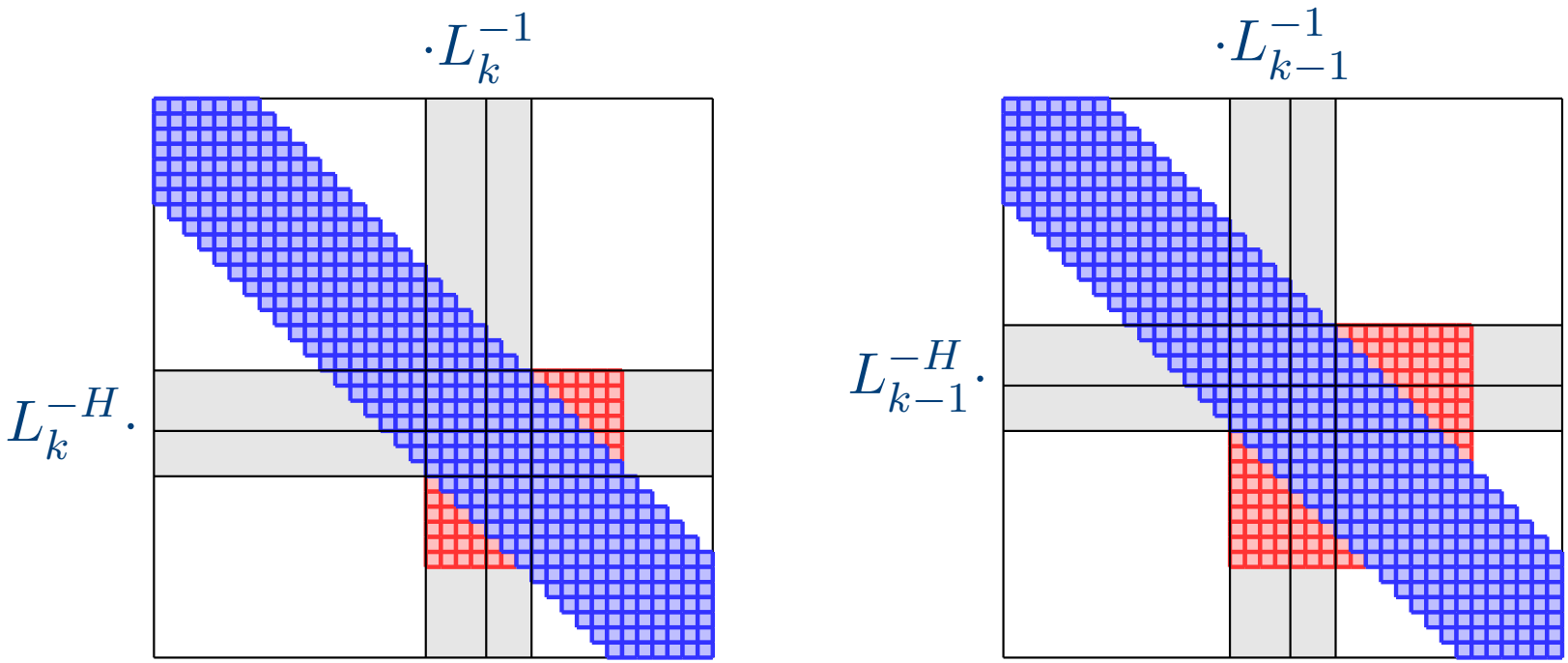
Band-preserving reduction III



This generates **fill** of size $b_B + n_b - 1$.

If one continues applying L_{k-1}^{-1} (and others) then ...

Band-preserving reduction III



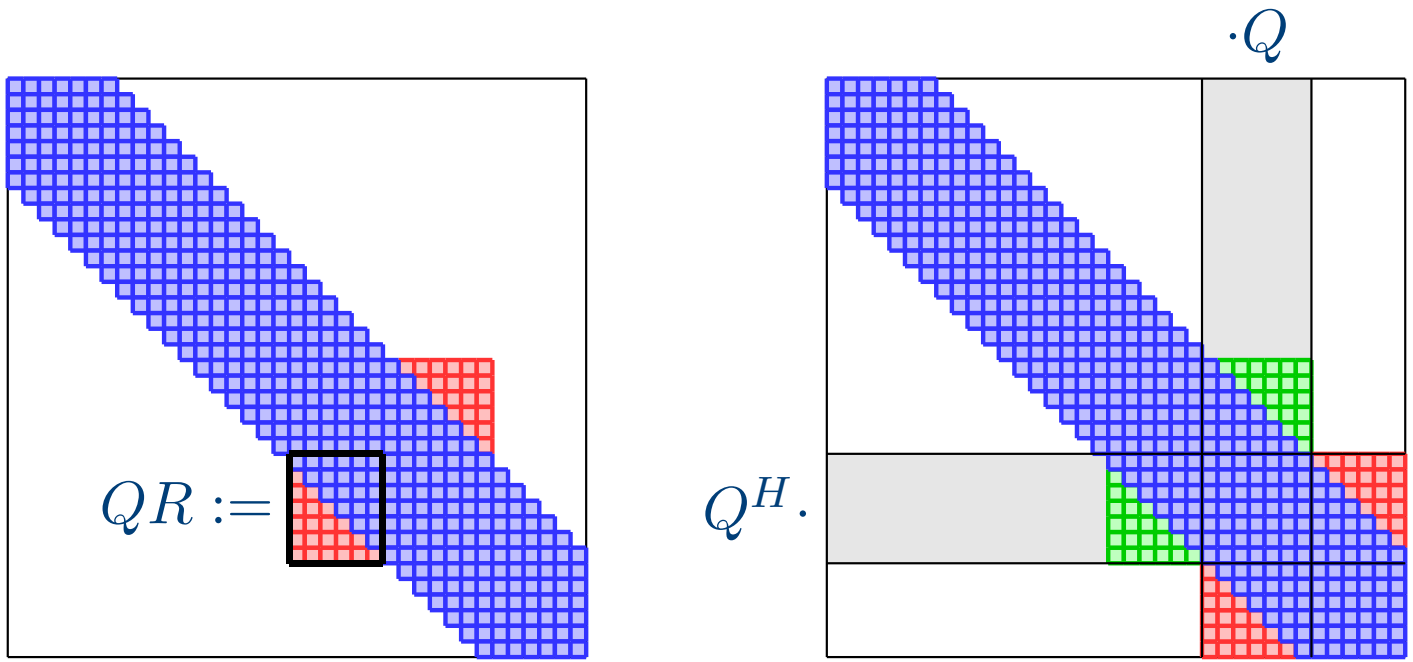
This generates **fill** of size $b_B + n_b - 1$.

If one continues applying L_{k-1}^{-1} (and others) then ...

... the fill grows by n_b rows and columns each time

Band-preserving reduction IV

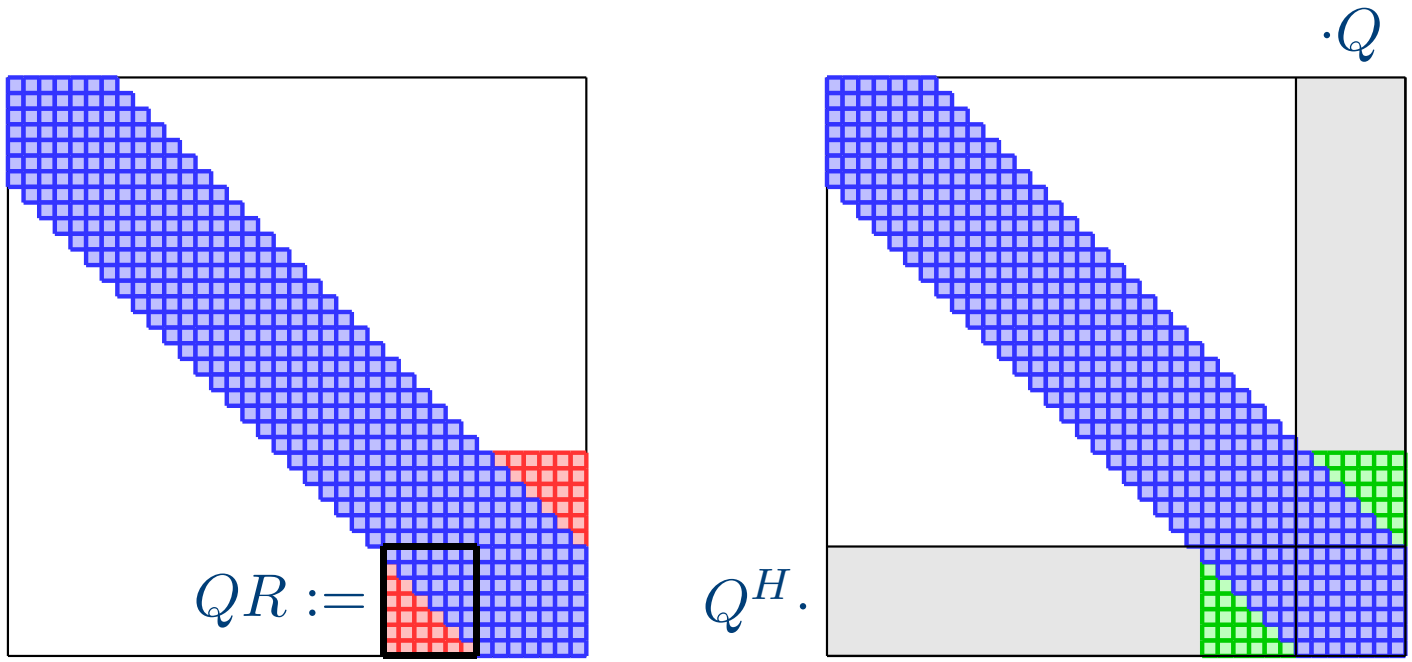
Therefore: Use a suitable (unitary) transformation to zero out the fill:



$\Rightarrow$ new fill b_A positions further down

Band-preserving reduction V

$\Rightarrow$ Repeat ...



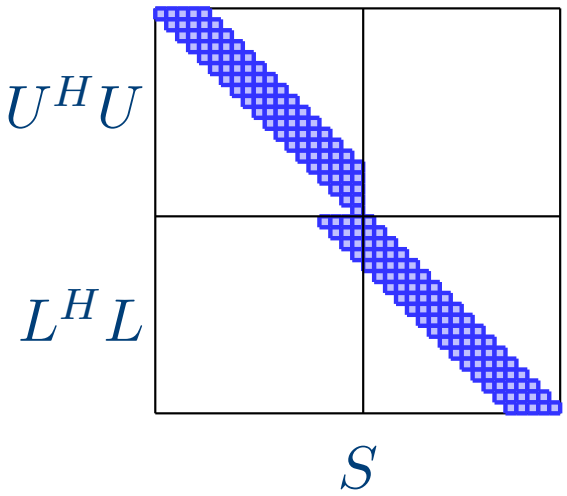
... until the fill vanishes at the end of the band ("bulge chasing")

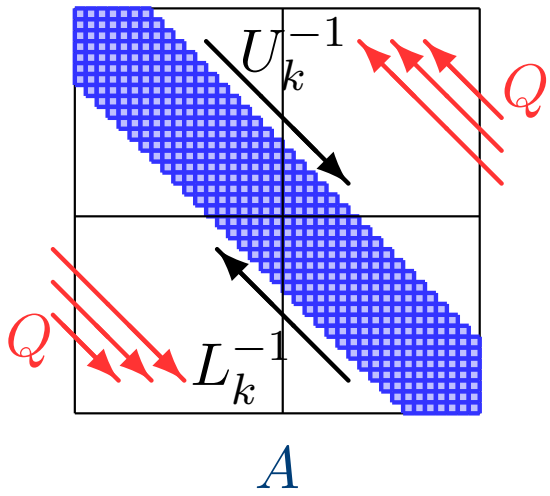
- Then continue with L_{k-1}^{-1}

Using twisted factorizations

$$B = S^H S,$$

where





- "Lower half": Apply the L_k^{-1} bottom–up, bulge chasing to the bottom (nearer) end of the band
- "Upper half": Apply the U_k^{-1} top–down, bulge chasing to the top (nearer) end of the band

$\Rightarrow$ Cost for bulge chasing roughly halved

Parallelization

[M. Rippl (TUM-SC) / MPCDF]

- Pipelining of successive chasing sweeps
- (For larger bandwidths aditionally)
  Parallelization of individual block operations

Outline

Updated API I

[MPCDM]

(Some features had been introduced before the latest R2018.05)

- Single/double precision available simultaneously
- New GEP driver
  - Allows specifying "same B"
- Versatile parameter list and reporting of statistics for manual control
- Autotuning capability for run-time parameters etc., e.g.,
  - Number of OpenMP threads (per routine)
  - Offload work to GPU ? (Per routine)
  - Blocking in back transformations

Updated API II

Simple example for autotuning:

```
autotune_handle = elpa_autotune_setup( handle, ELPA_AUTOTUNE_FAST, \\
                                       ELPA_AUTOTUNE_DOMAIN_REAL, \\
                                       &error ) ;

for ( i=0, i<20, i++ ) {

  unfinished = elpa_autotune_step( handle, autotune_handle ) ;

  if ( unfinished == 0 )
    printf( "ELPA autotuning finished in the %d th SCF step\n", i ) ;

  /* Solve EV problem */
  elpa_eigenvectors( handle, a, ev, z, &error ) ;

}

elpa_autotune_set_best( handle, autotune_handle ) ;

elpa_autotune_deallocate( autotune_handle ) ;
```

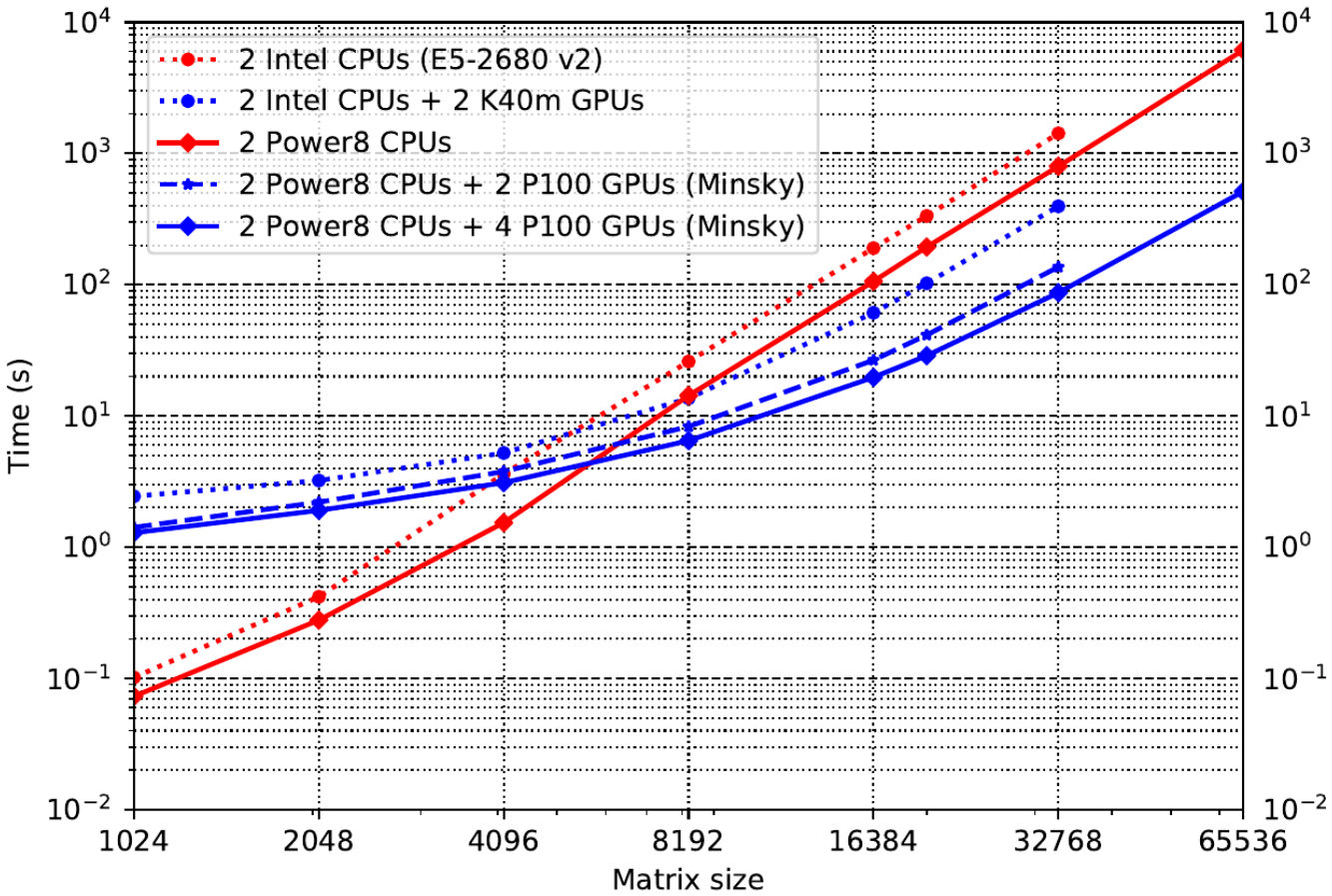
Optimization and porting to new architectures I

[MPCDM]

- Intel KNL
  - Scaling up to ~ 200.000 cores demonstrated on ANL's Theta
  - Run with 1048k matrix
- (ongoing) Optimization for ARM 64bit
- Intel Skylake: AVX-2 $\rightarrow$ AVX-512: ~ 1.6 speed-up
- Porting to GPUs: > 10 speed-up possible
  (depending on matrix size)

Optimization and porting to new architectures II

Complete (real, double) SEP



Outline

Recent/upcoming developments

[ELPA-AEO consortium]

- Enhanced autotuning capabilities
  (available)
- Performance optimization (e.g., GPUs and AVX-512)
  (ongoing, immediately made available to the users)
- Improved performance of the reduction generalized to standard for full matrices
  (expected soon)
- New reduction generalized to standard for banded matrices
  (under development)