Solvers and Eigensolvers for Multicore Processors

Paolo Bientinesi

AICES, RWTH Aachen
pauldj@aices.rwth-aachen.de

Max-Plank-Institute für biologische Kybernetik
March 18th, 2011
Tübingen, Germany

Dense Linear Algebra Kernels

- Matrix-matrix multiplication (GEMM): $C \leftarrow C + AB$

- Factorizations: $LU = A$, $LL^T = A$, $QR = A$
- Linear system: $AX = B$
- Transformations: $QAQ^T = T$, $QA = H$
- Matrix Equations: $AX + XB = C$

- Eigenproblems: $AZ = Z\Lambda$
- Generalized eigenproblems: $AZ = BZ\Lambda$
- ...

Dense Linear Algebra Kernels

RWTH AACHEN UNIVERSITY

- Matrix-matrix multiplication (GEMM): $C \leftarrow C + AB$

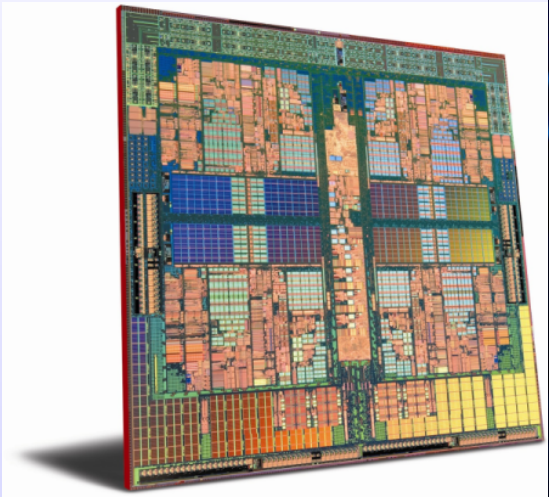
- Factorizations: $LU = A,\ LL^T = A,\ QR = A$
- Linear system: $AX = B$
- Transformations: $QAQ^T = T,\ QA = H$
- Matrix Equations: $AX + XB = C$

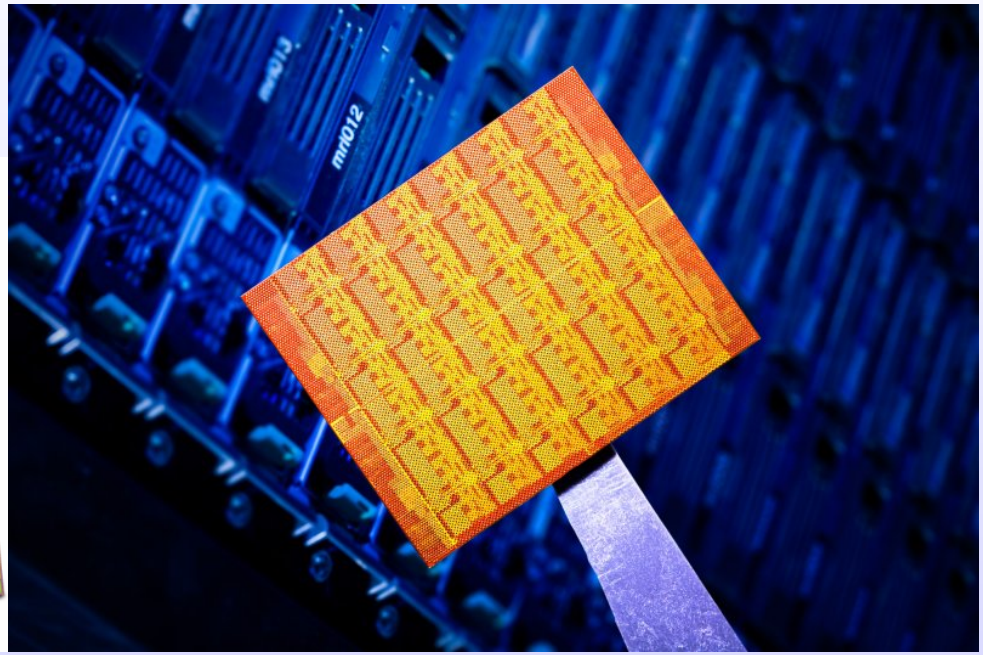
- Eigenproblems: $AZ = Z\Lambda$
- Generalized eigenproblems: $AZ = BZ\Lambda$
- ...

Objective: High-performance

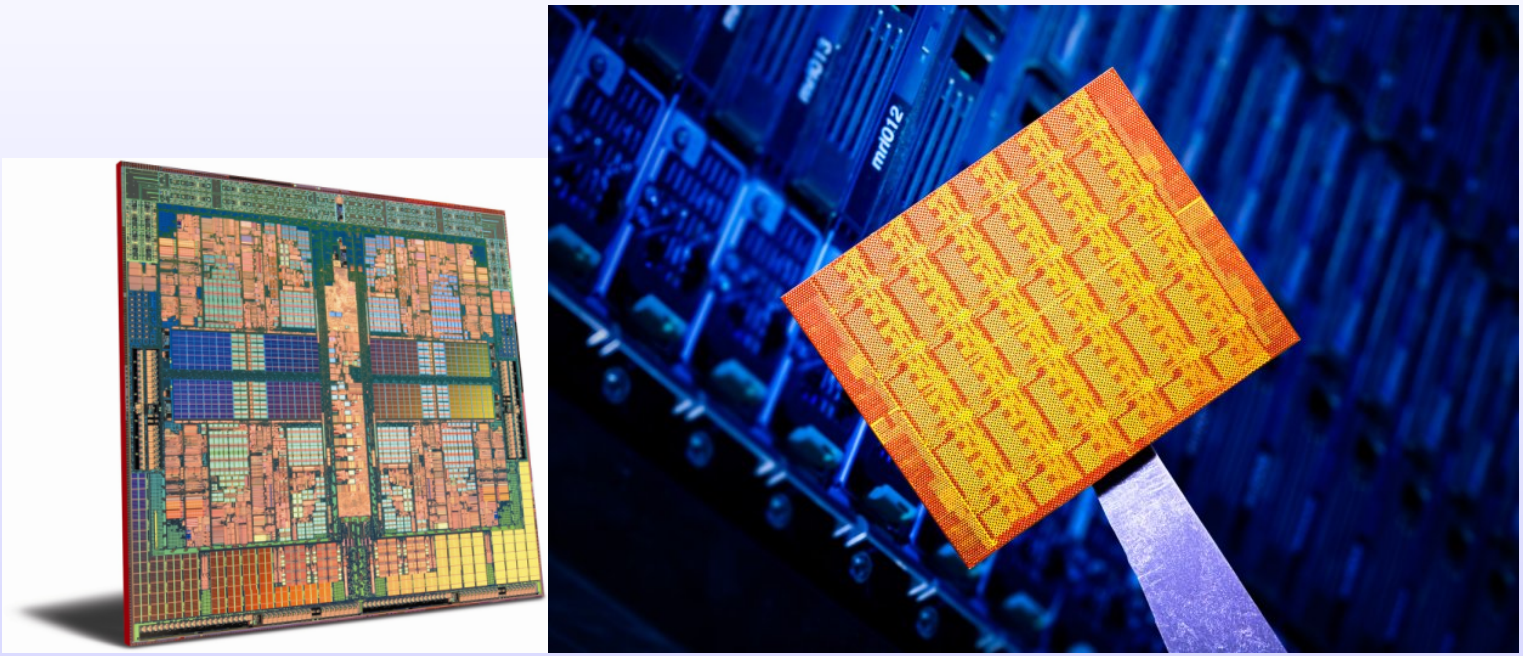
- Numerical stability
- Multiple algorithmic variants
- Multiple implementations
- Multicore $\rightarrow$ Multiple types of parallelism!

AICES





Multi-cores invasion: 499/500 entries of the Top 500



Multi-cores invasion: 499/500 entries of the Top 500

4 cores, 8 cores, . . . 24 cores, . . .

- More parallelism than we know what to do with?
- Is multi-threaded BLAS the solution for LA libs?
- Linear solvers $\neq$ Eigensolvers

Linear Algebra: Modularity!

Algorithms expressed in terms of simpler linear algebra operations

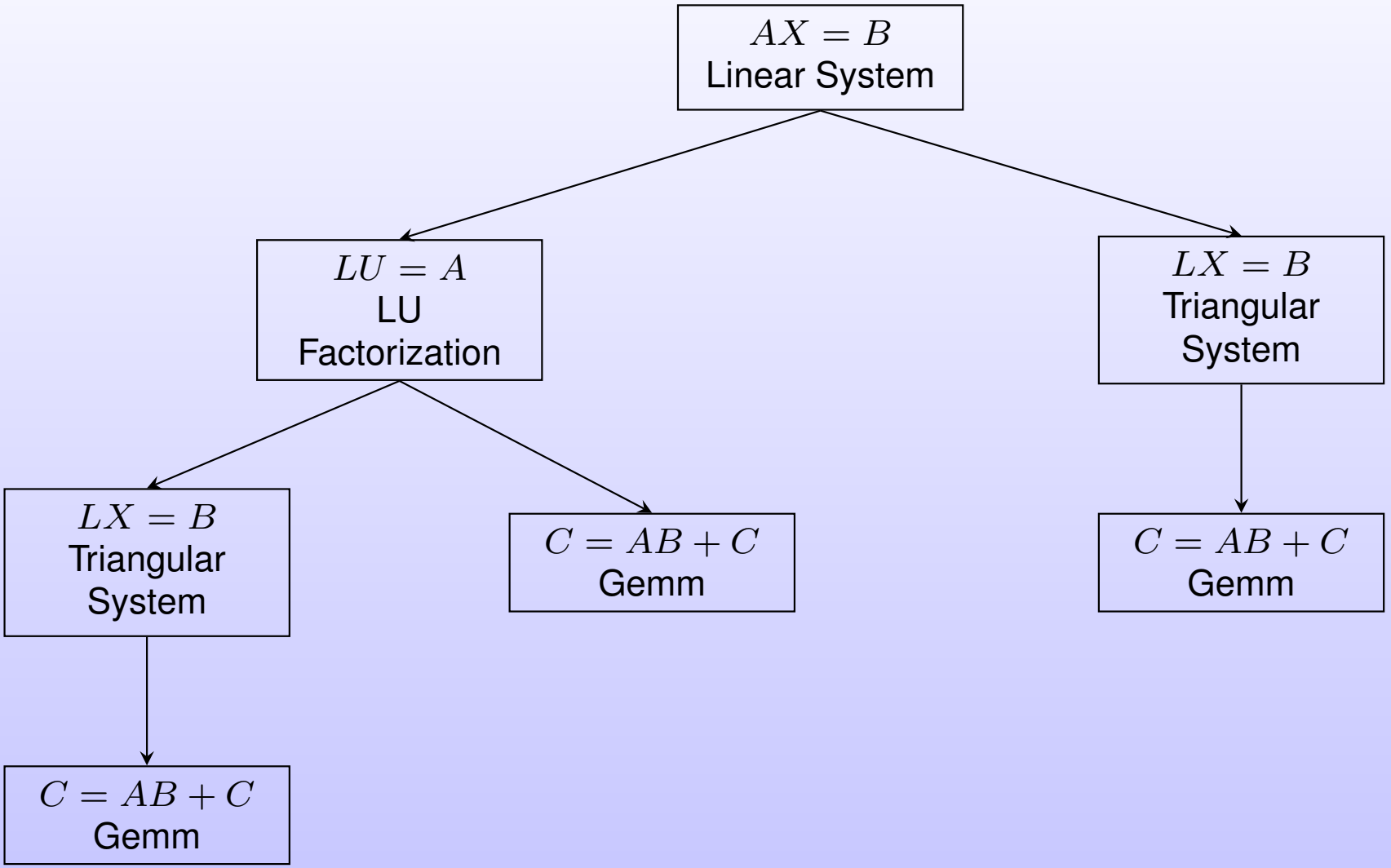
Algorithms expressed in terms of simpler linear algebra operations

BLAS: Basic Linear Algebra Subroutines

BLAS-1:	$y := y + \alpha x$ $\beta := \alpha + x^T y$	$x, y \in \mathbb{R}^n$
BLAS-2:	$y := y + Ax$ $y := A^{-1}x$	$A \in R^{n \times n},\ x, y \in R^n$
BLAS-3:	$C := C + AB$ $C := A^{-1}B$	$A, B, C \in R^{n \times n}$

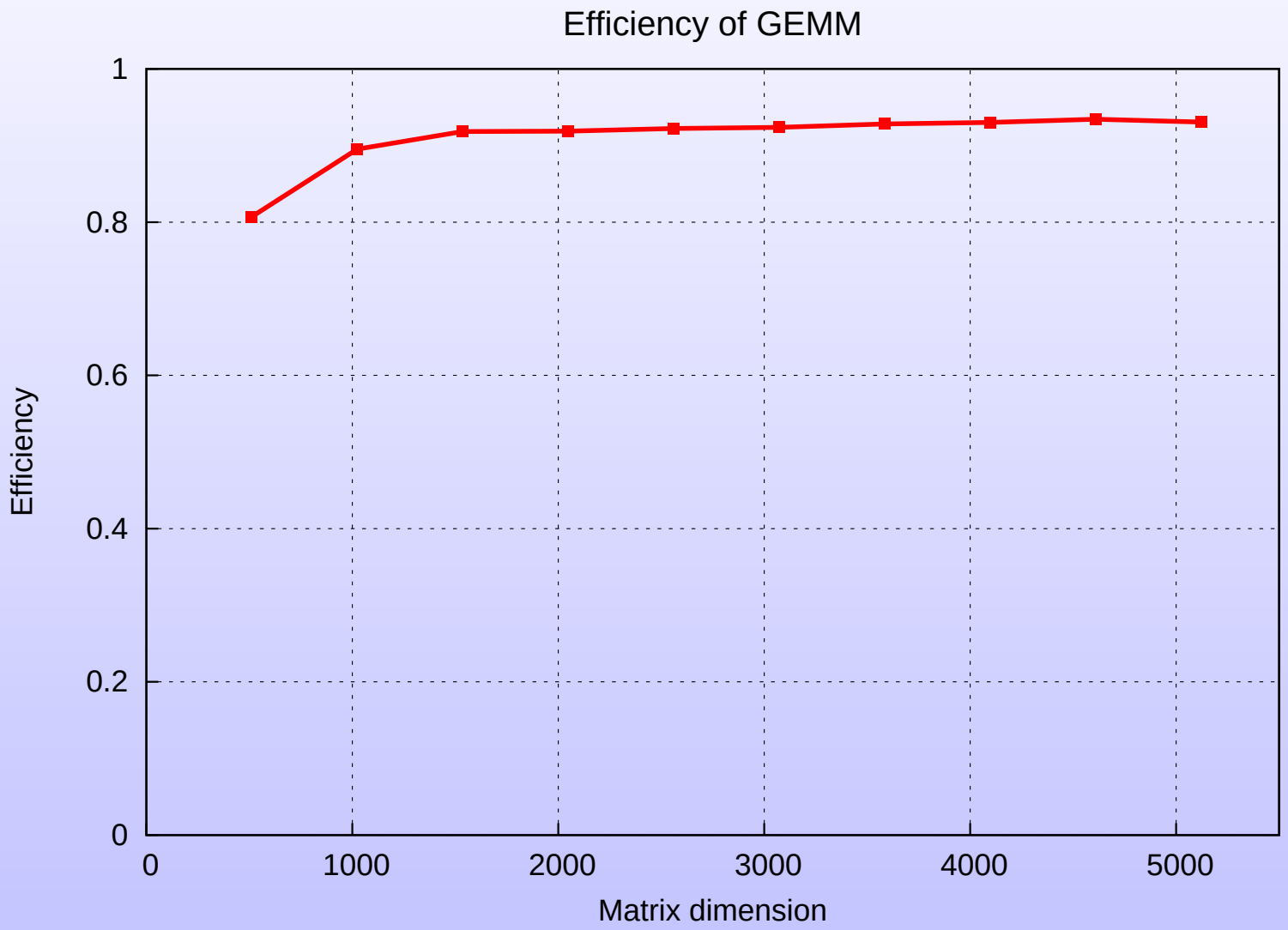
ESSL (IBM), MKL (Intel), ATLAS, GotoBLAS, ...

Example: $AX = B$


$AX = B$
Linear System
$LU = A$
LU
Factorization
$LX = B$
Triangular
System
$LX = B$
Triangular
System
$C = AB + C$
Gemm
$C = AB + C$
Gemm
$C = AB + C$
Gemm

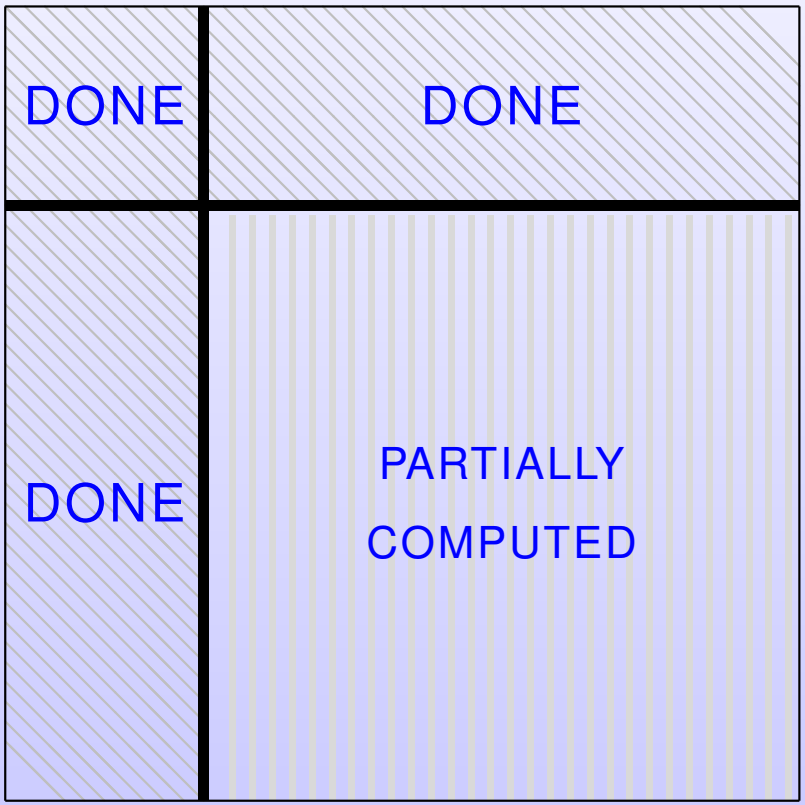
Performance of BLAS

Single threaded GEMM



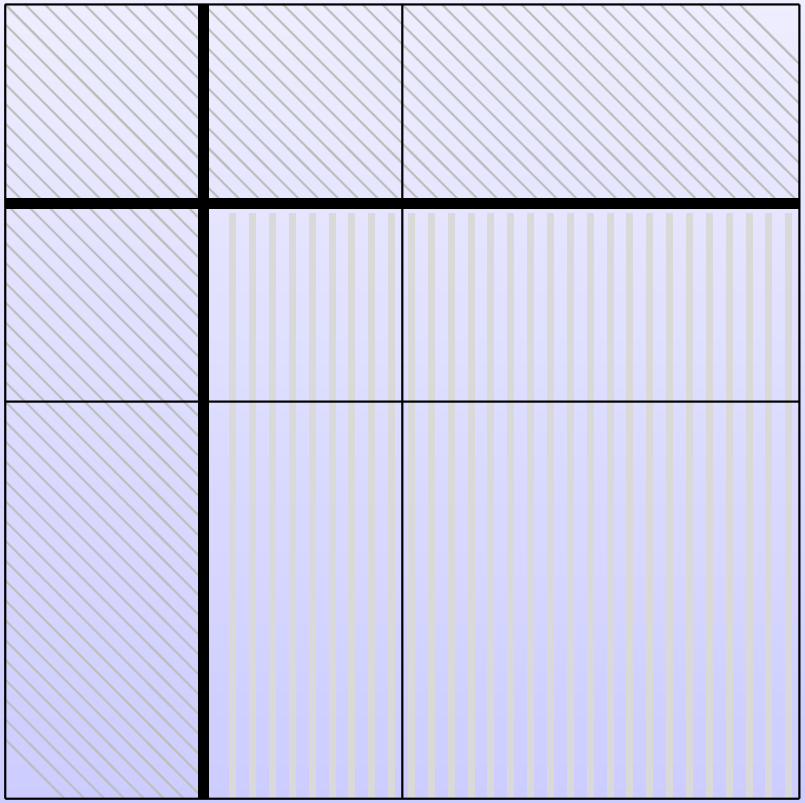
LU factorization: loop-based algorithm

Iteration i: completed



LU factorization: loop-based algorithm

Iteration i+1: repartitioning

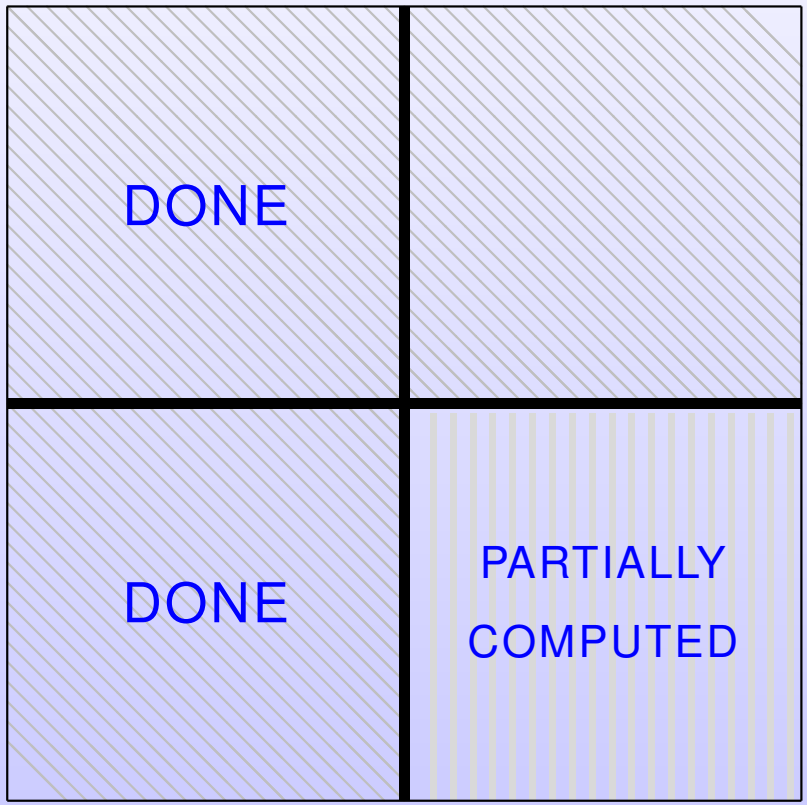


LU factorization: loop-based algorithm

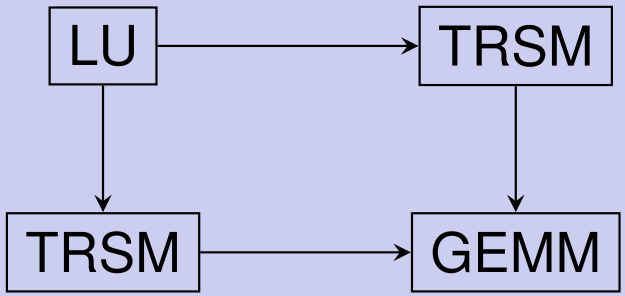
Iteration i+1: computation

LU factorization: loop-based algorithm

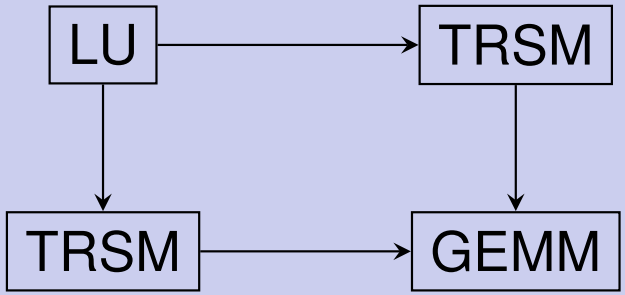
Iteration i+1: completed (boundary shift)



Solution #1: Multithreaded BLAS



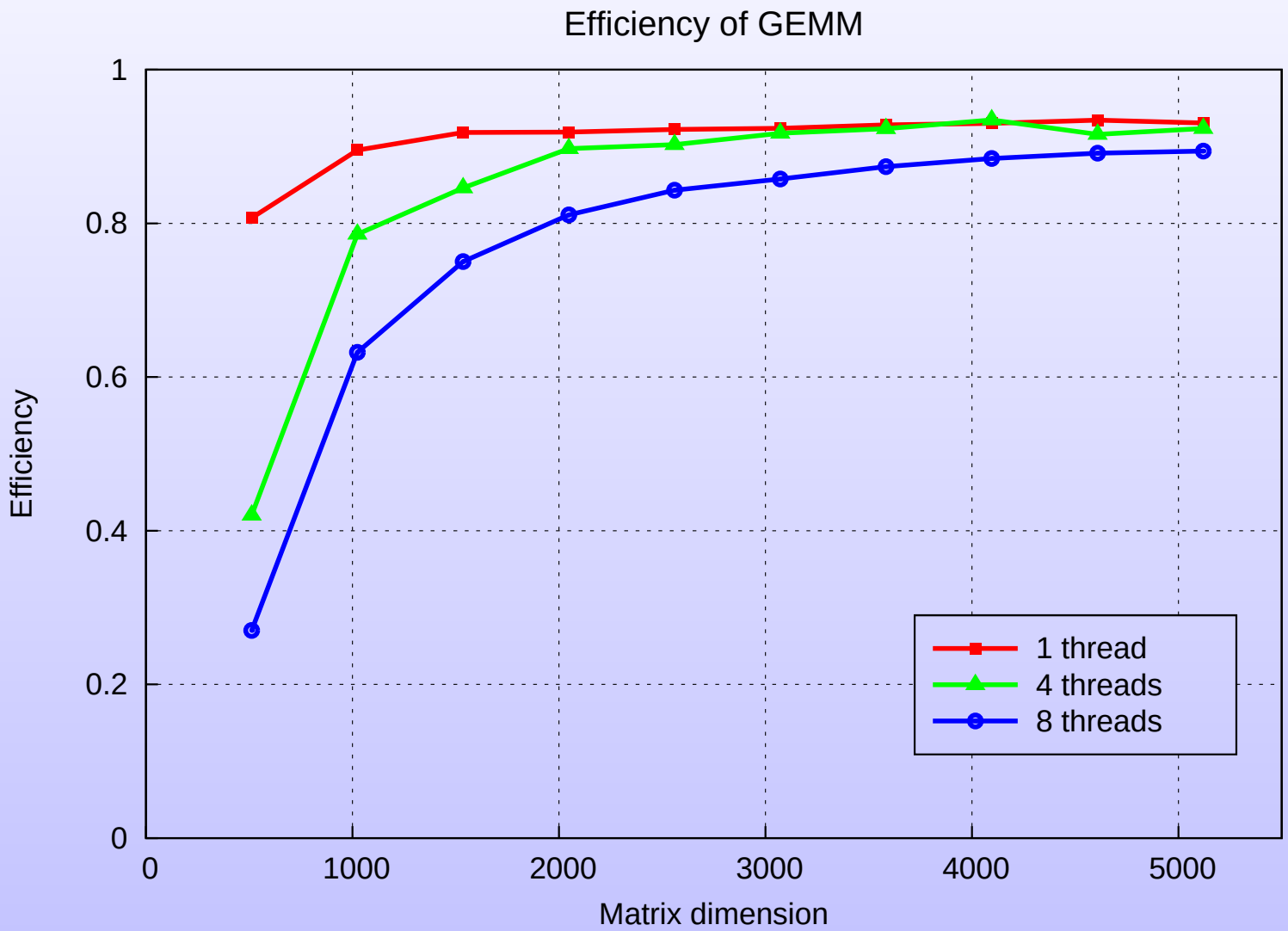
Solution #1: Multithreaded BLAS



- Advantage: ease of use. Legacy code!
- Drawback: synchronization.

Performance of BLAS

Multithreaded GEMM

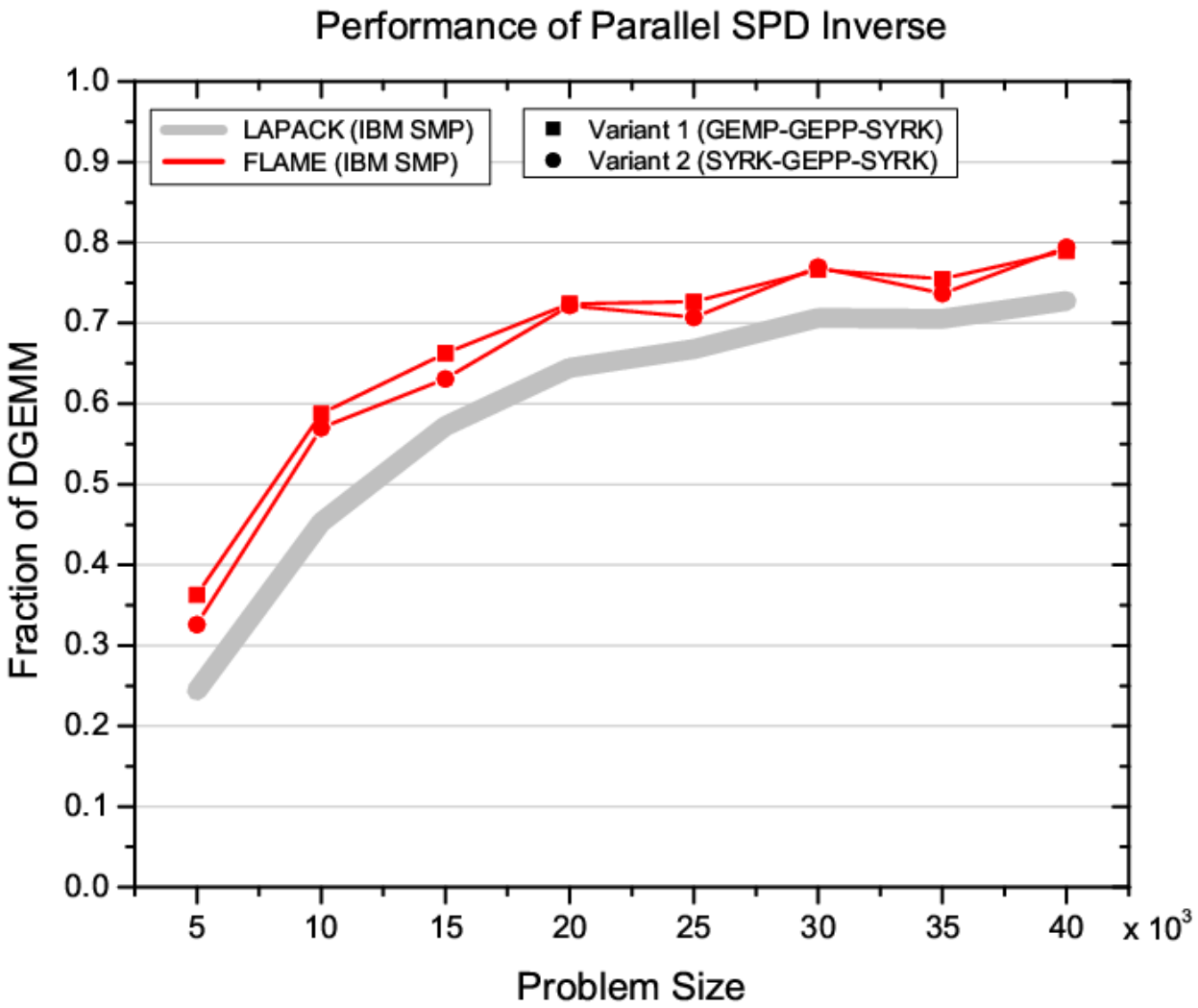


Example: SPD Inverse

- Inversion of a Symmetric Positive Definite matrix
- Covariance matrix
- Very large dense problems

Cholesky factorization
↓
Triangular inversion
↓
Matrix-matrix multiplication

Example: SPD Inverse



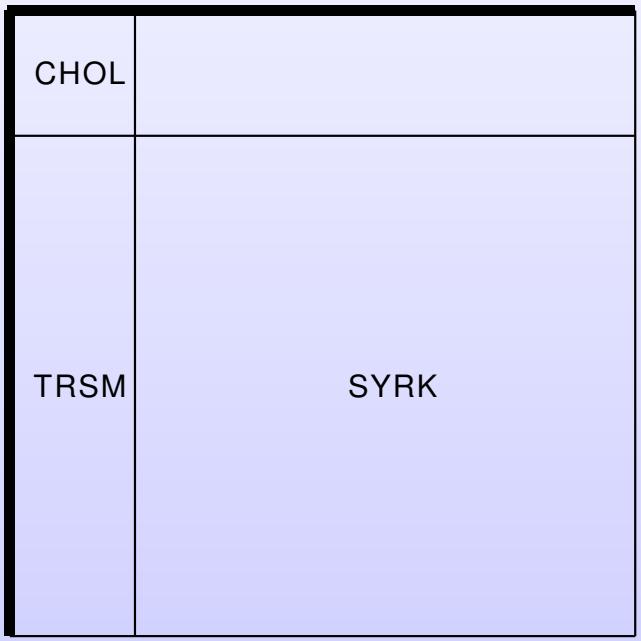
Parallelism: can we do better?

Solution #2: Algorithms by blocks

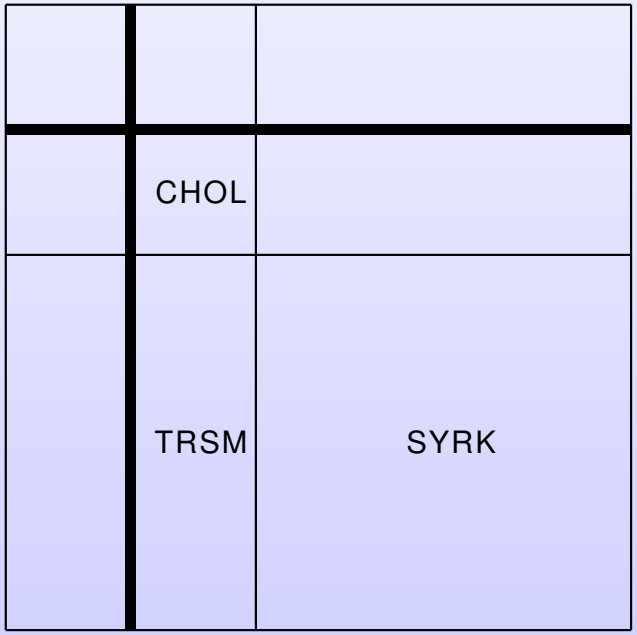
- Advantage: out of order execution.
- Advantage: parallelism limited only by the data dependencies between operations.
- Drawback: plateux.

Cholesky factorization

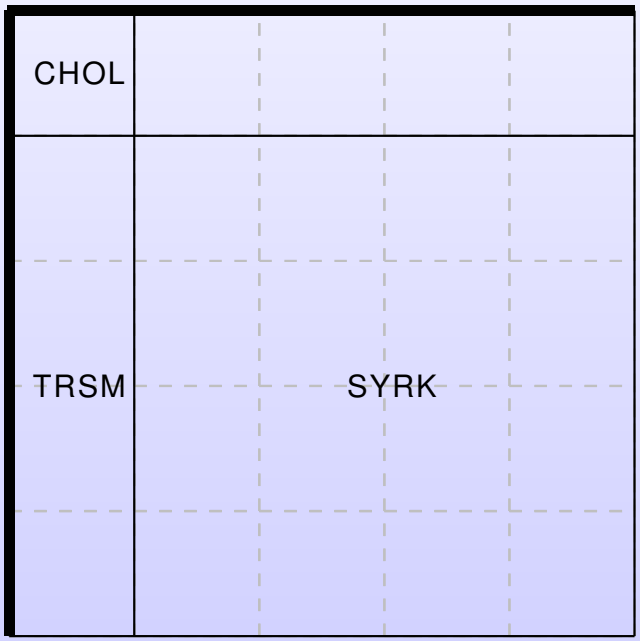
$LL^T = A$



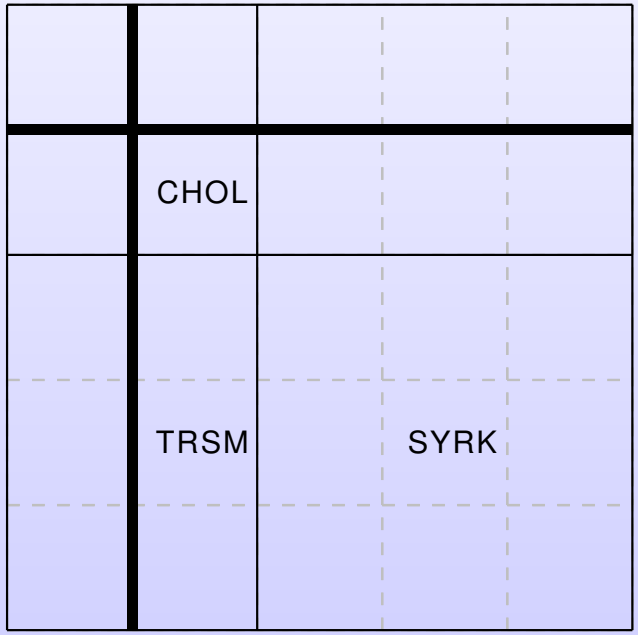
Iteration 1



Iteration 2



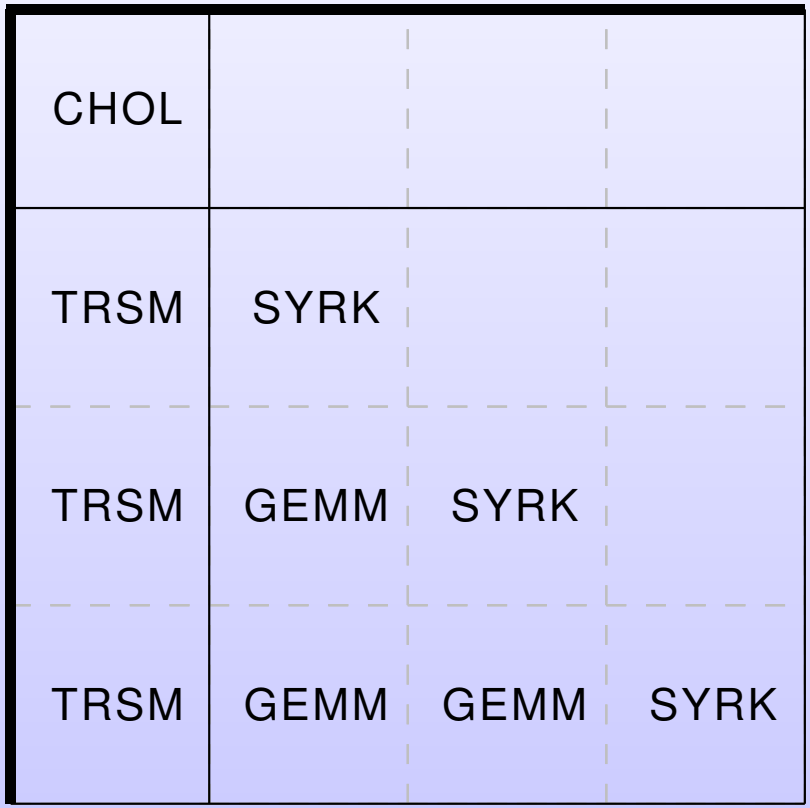
Iteration 1



Iteration 2

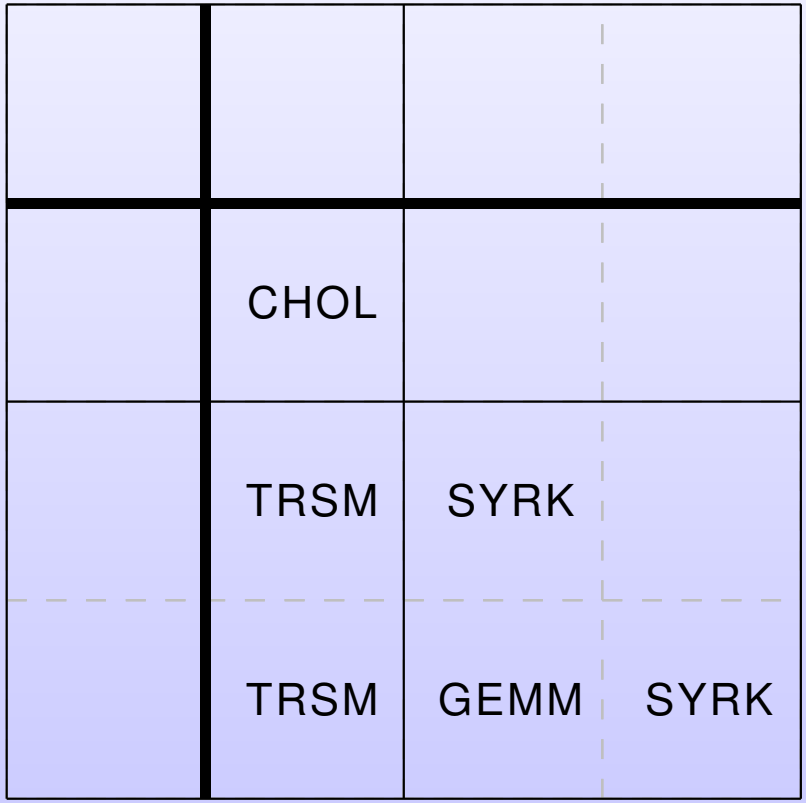
Decomposing the computation

Iteration 1



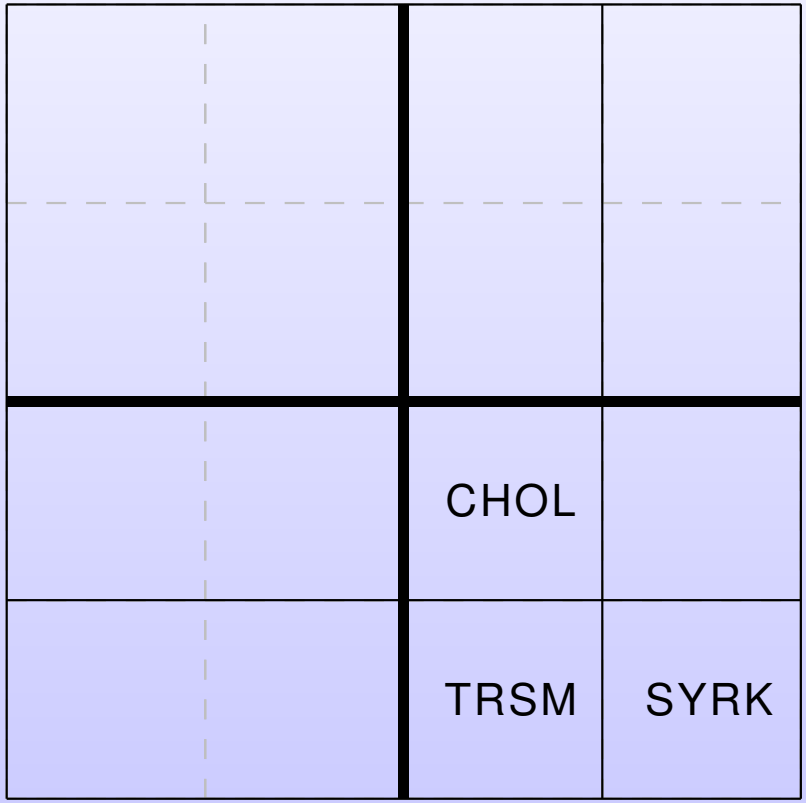
Decomposing the computation

Iteration 2



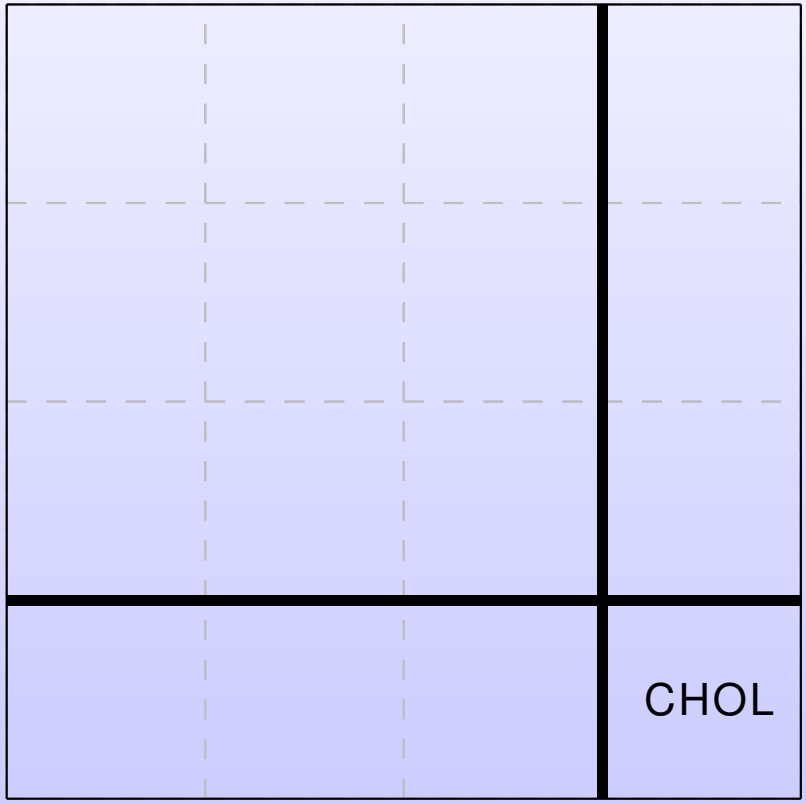
Decomposing the computation

Iteration 3

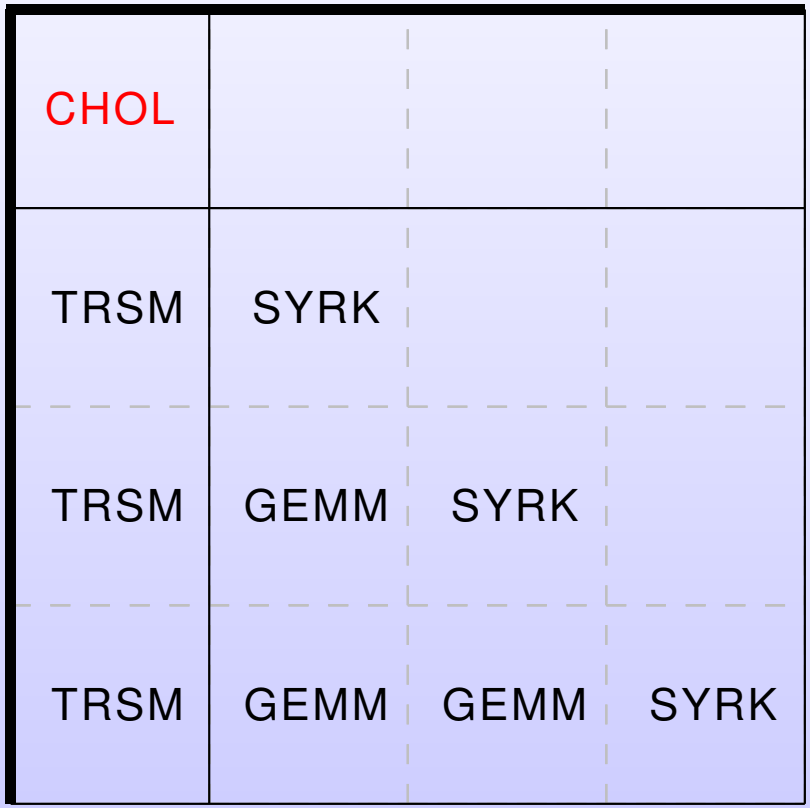


Decomposing the computation

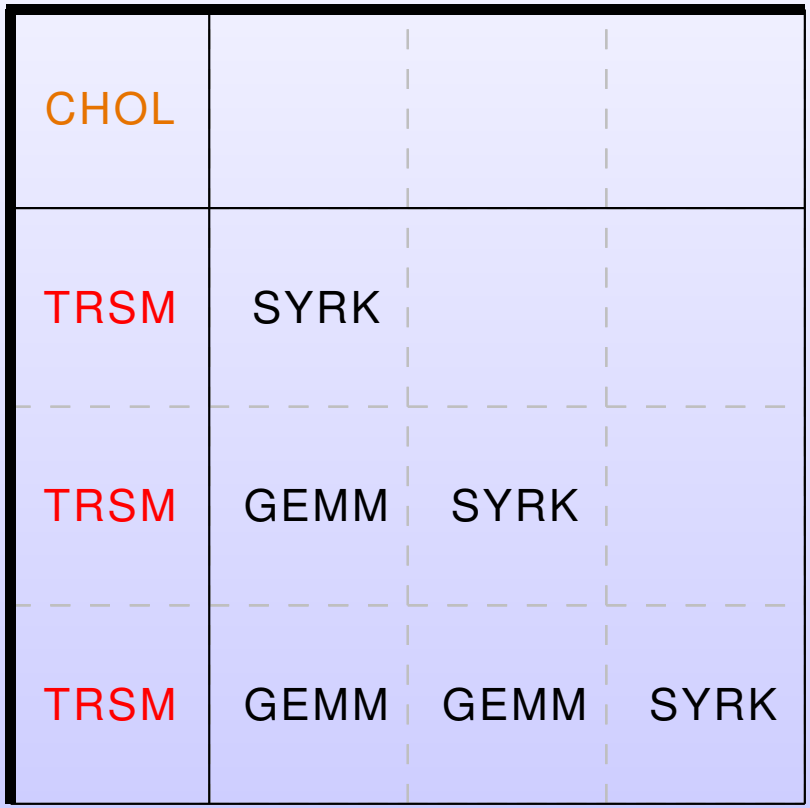
Iteration 4



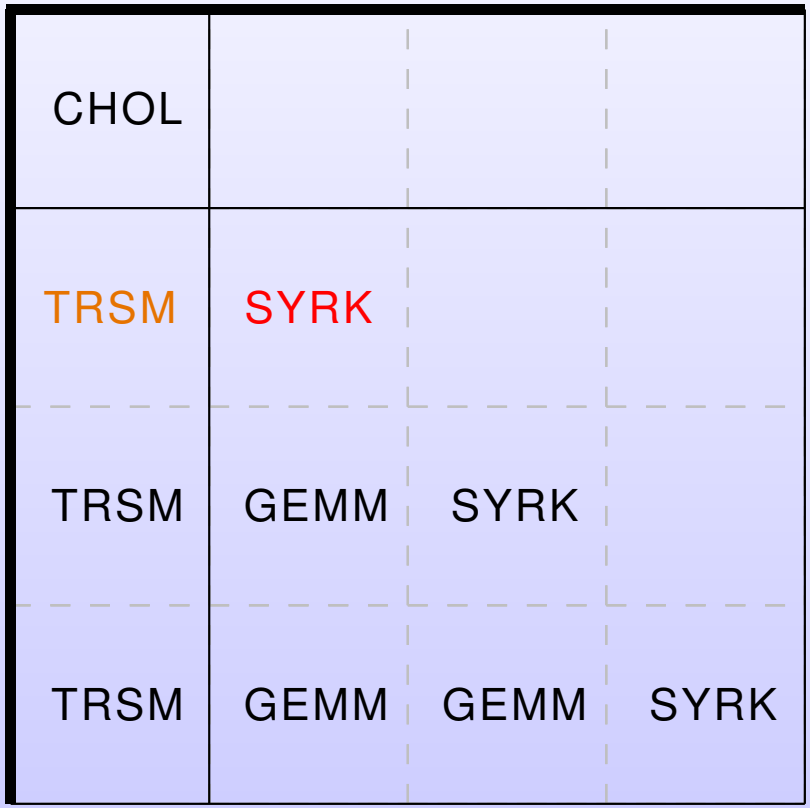
Dependencies



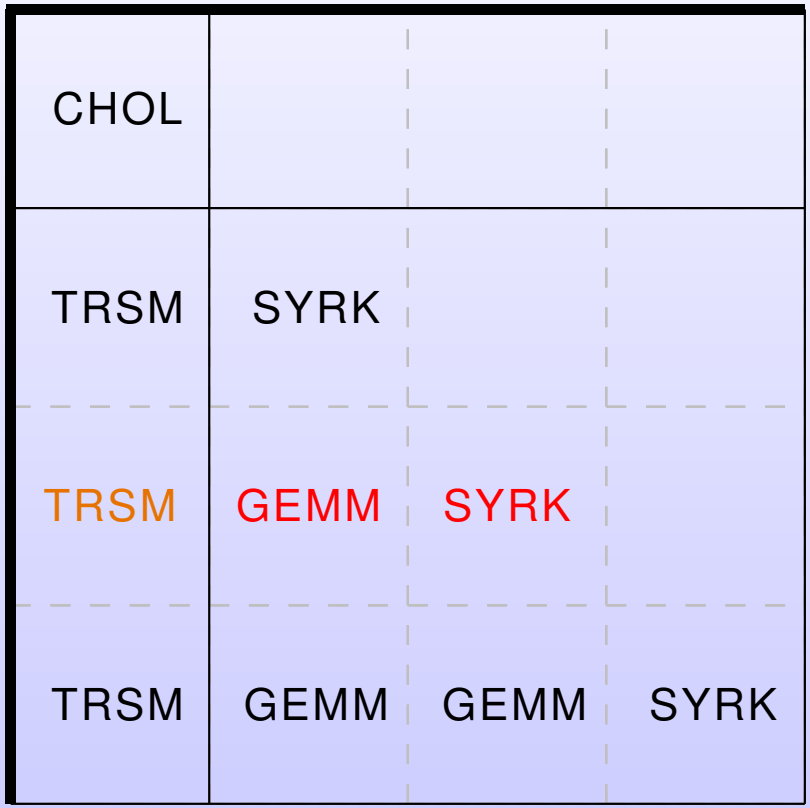
Dependencies



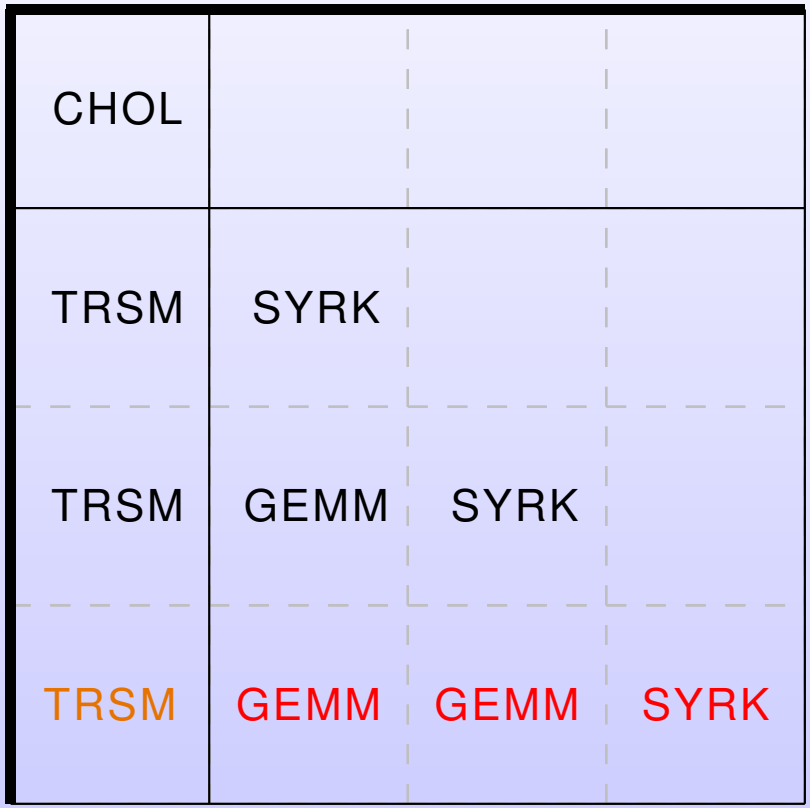
Dependencies



Dependencies

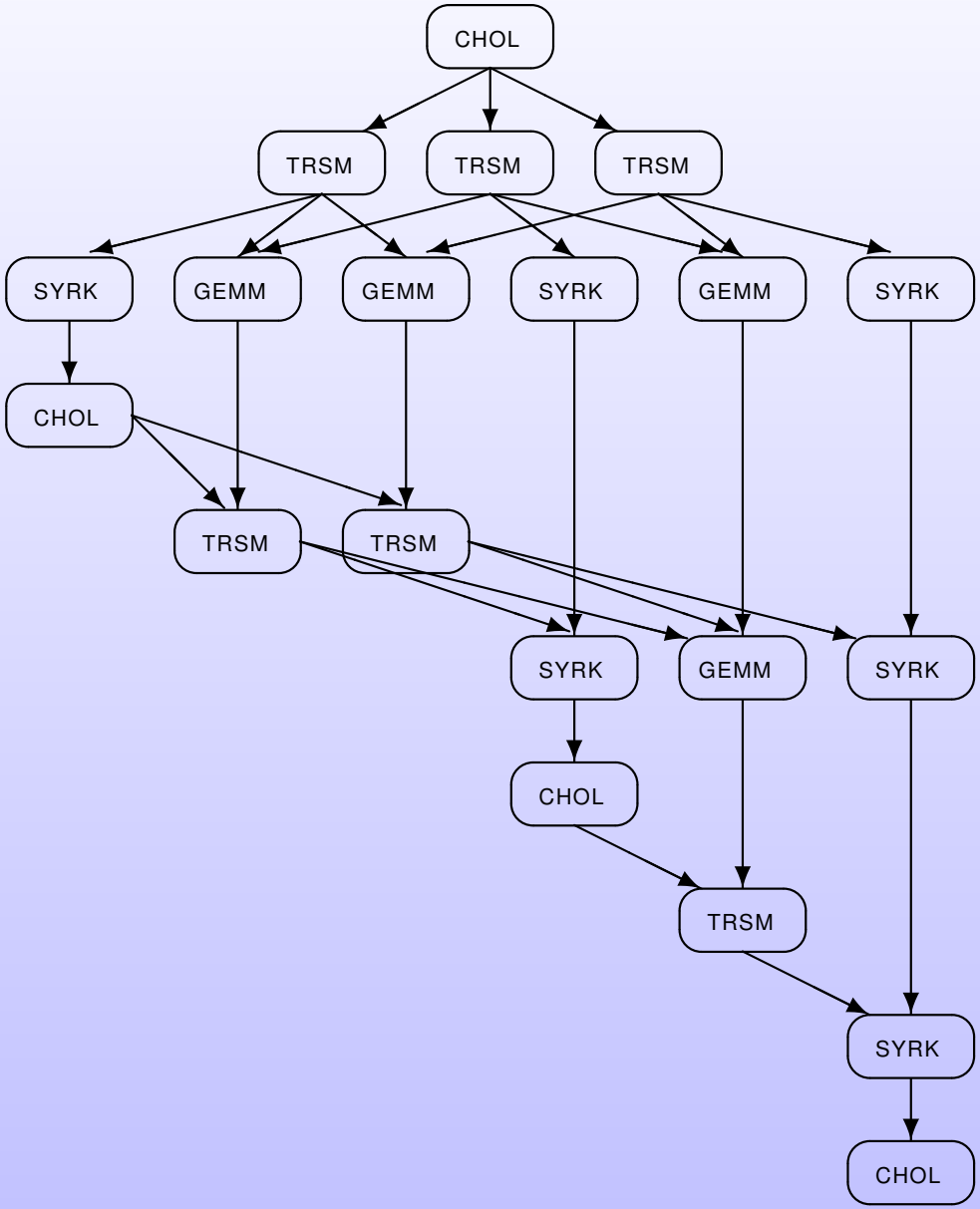


Dependencies



DAG - Dependencies

4×4-tile matrix



Task Execution

4×4-tile matrix

Stage	Scheduled Tasks			
1	CHOL			
2	TRSM	TRSM	TRSM	TRSM
3	SYRK	GEMM	SYRK	GEMM
4	GEMM	SYRK	GEMM	GEMM
5	GEMM	SYRK	CHOL	
6	TRSM	TRSM	TRSM	
7	SYRK	GEMM	SYRK	GEMM
8	GEMM	SYRK	CHOL	
9	TRSM	TRSM		
10	SYRK	GEMM	SYRK	
11	CHOL			
12	TRSM			
13	SYRK			
14	CHOL			

SPD Inverse again: Chol+Inv+GEMM

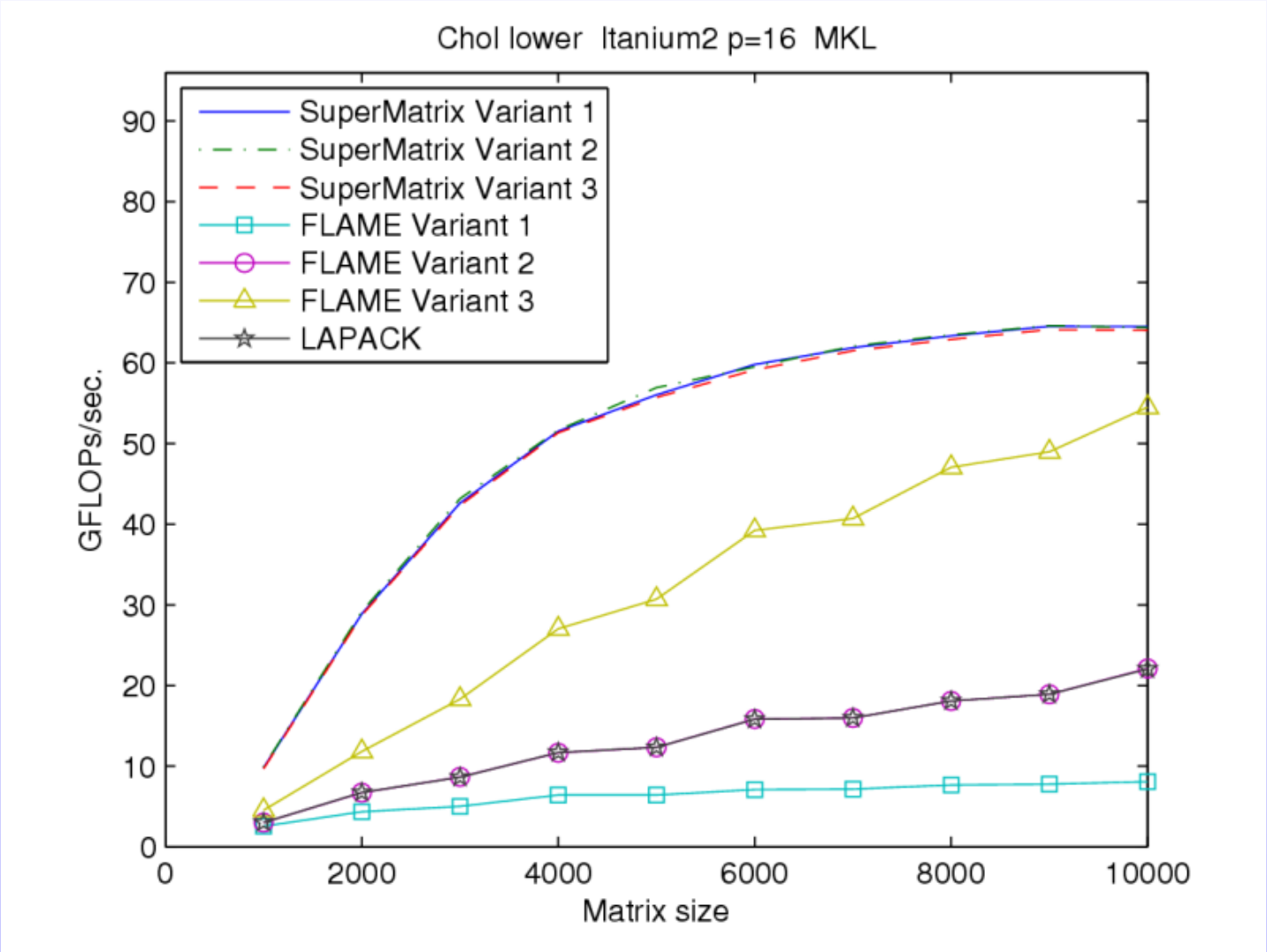
5×5-tile matrix

SPD Inverse again: Chol+Inv+GEMM

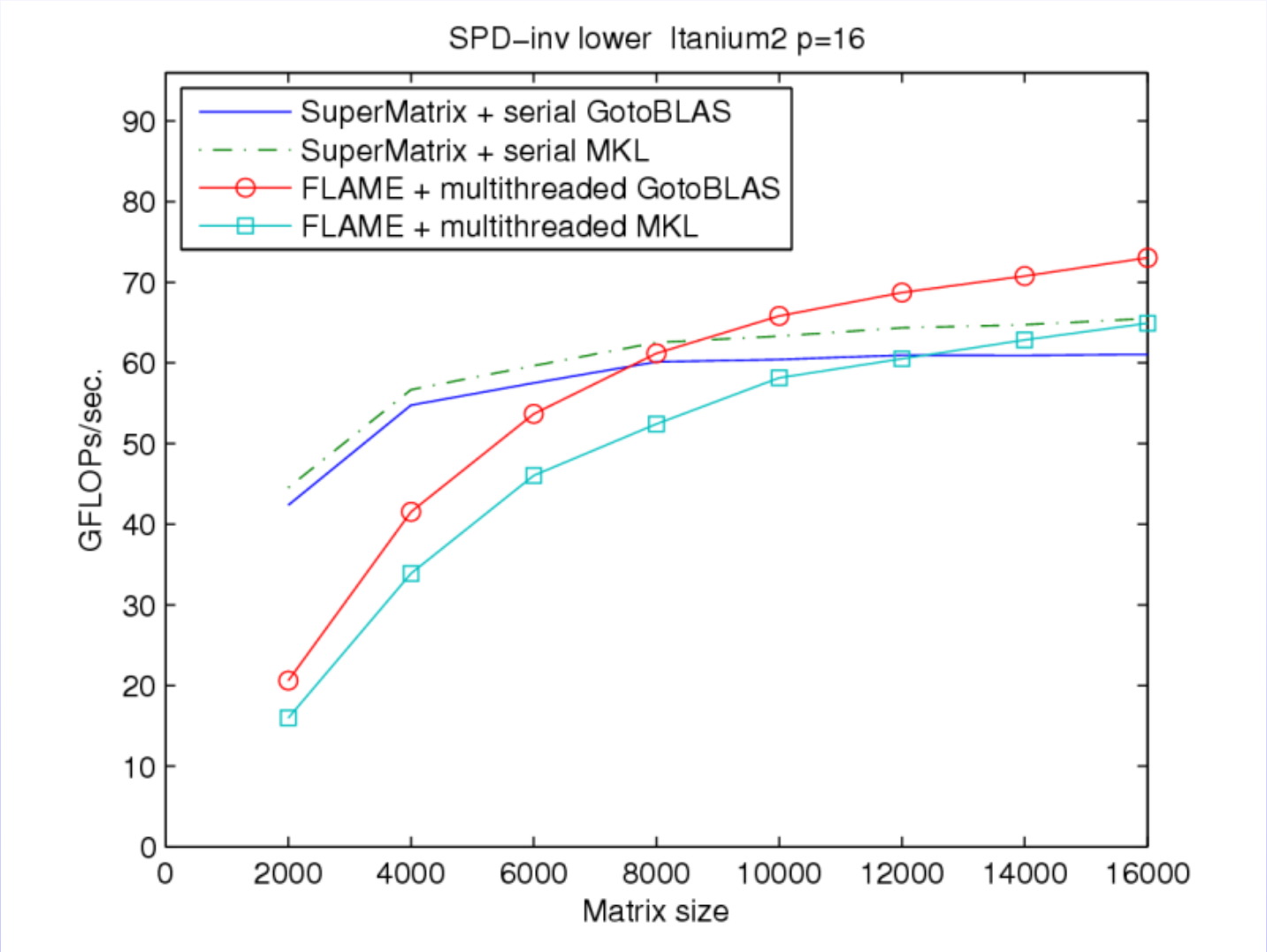
5×5-tile matrix

Stage	Scheduled Tasks			
1	CHOL			
2	TRSM	TRSM	TRSM	TRSM
3	SYRK	GEMM	SYRK	GEMM
4	GEMM	SYRK	GEMM	GEMM
5	GEMM	SYRK	CHOL	TRSM
6	TRSM	TRSM	TRSM	TRSM
7	TRSM	TRSM	TRINV	SYRK
8	GEMM	SYRK	GEMM	GEMM
9	SYRK	TTMM	CHOL	TRSM
10	TRSM	TRSM	TRSM	TRSM
11	GEMM	GEMM	GEMM	SYRK
12	GEMM	SYRK	TRSM	CHOL
13	TRSM	TRSM	TRINV	SYRK
14	TRSM	GEMM	GEMM	GEMM
15	GEMM	TRMM	SYRK	TRSM
16	TRSM	TTMM	CHOL	TRSM
17	SYRK	TRINV	GEMM	SYRK
18	GEMM	GEMM	GEMM	TRMM
19	TRMM	TRSM	TRSM	TRSM
20	TRSM	TRSM	TRSM	TRSM
21	TTMM	SYRK	GEMM	SYRK
22	TRINV	GEMM	GEMM	TRINV
23	SYRK	SYRK	GEMM	SYRK
24	TRMM	GEMM	TRMM	GEMM
25	TRMM	SYRK	GEMM	GEMM
26	TTMM	GEMM	TRMM	TRMM
27	SYRK	TRMM		
28	TRMM			
29	TTMM			

Cholesky, algorithm by block



Multithreaded vs. algorithm by block



Multithreaded BLAS vs. Algorithms by blocks

No absolute winner: crossover!

- ✓ Ease of use
- ✖ Synchronization
- ✓ Out of order execution
- ✓ Parallelism dictated by data dependencies
- ✖ Plateux

$AX = X\Lambda$

- Input: $A \in \mathcal{C}^{n \times n}$, $A^H = A$; #eigenpairs: $1 \leq k \leq n$
- Output: $X \in \mathcal{C}^{n \times k}$ eigenvectors
  $\Lambda \in \mathcal{R}^{k \times k}$ eigenvalues

RWTH AACHEN UNIVERSITY

$AX = X\Lambda$

- Input: $A \in \mathcal{C}^{n \times n}$, $A^H = A$; #eigenpairs: $1 \leq k \leq n$
- Output: $X \in \mathcal{C}^{n \times k}$ eigenvectors
  $\Lambda \in \mathcal{R}^{k \times k}$ eigenvalues

Approach

$T = Q^H A Q$	Reduction to tridiagonal form	$O(n^3)$
$TZ = Z\Lambda$	Tridiagonal eigenproblem	$O(kn)$–$O(n^3)$
$X = QZ$	Backtransformation	$O(kn^2)$

AICES

Problem

$AX = X\Lambda$

- Input: $A \in \mathcal{C}^{n \times n}$, $A^H = A$; #eigenpairs: $1 \leq k \leq n$
- Output: $X \in \mathcal{C}^{n \times k}$ eigenvectors
  $\Lambda \in \mathcal{R}^{k \times k}$ eigenvalues

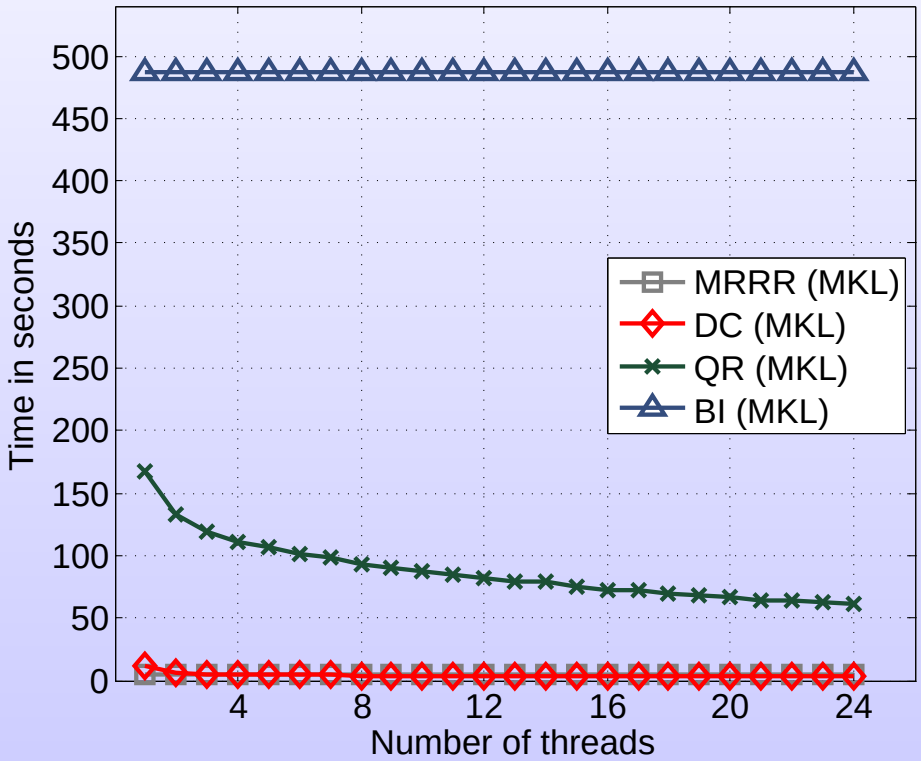
Approach

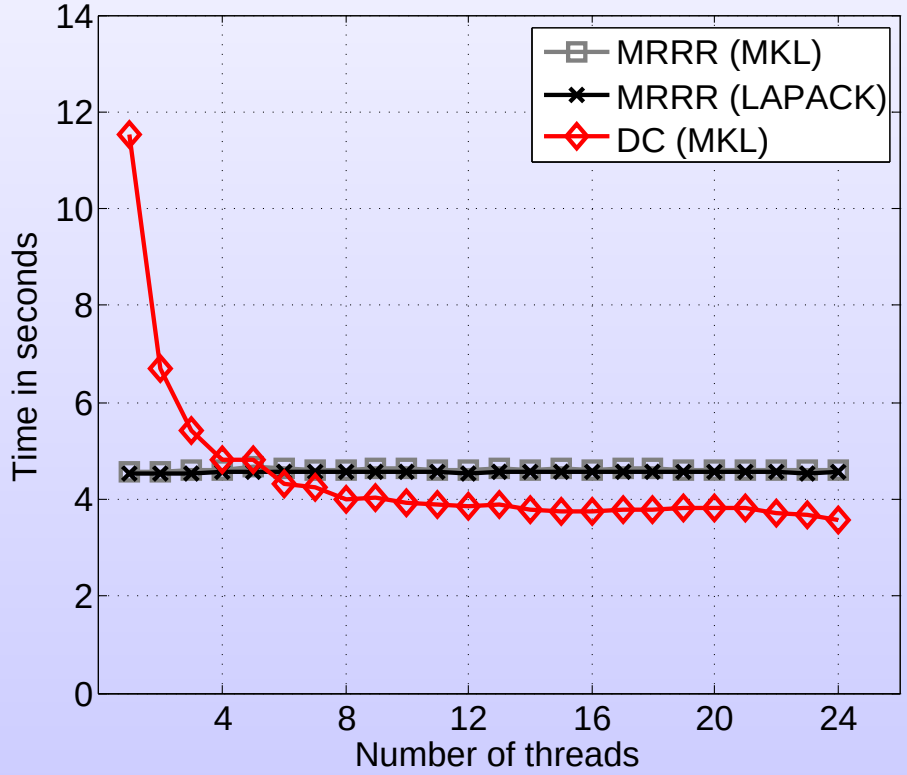
$T = Q^H A Q$	Reduction to tridiagonal form	$O(n^3)$
$TZ = Z\Lambda$	Tridiagonal eigenproblem	$O(kn)$–$O(n^3)$
$X = QZ$	Backtransformation	$O(kn^2)$

Algorithms

Inverse Iteration (1958): subsets	$O(kn^2)$
QR (1961): high-accuracy	$O(n^3)$
Divide & Conquer (1981): parallel, BLAS3	$O(n^3)$
MRRR (1997): subsets, no re-orth.	$\mathbf{O(kn)}$

Multi-threaded BLAS?





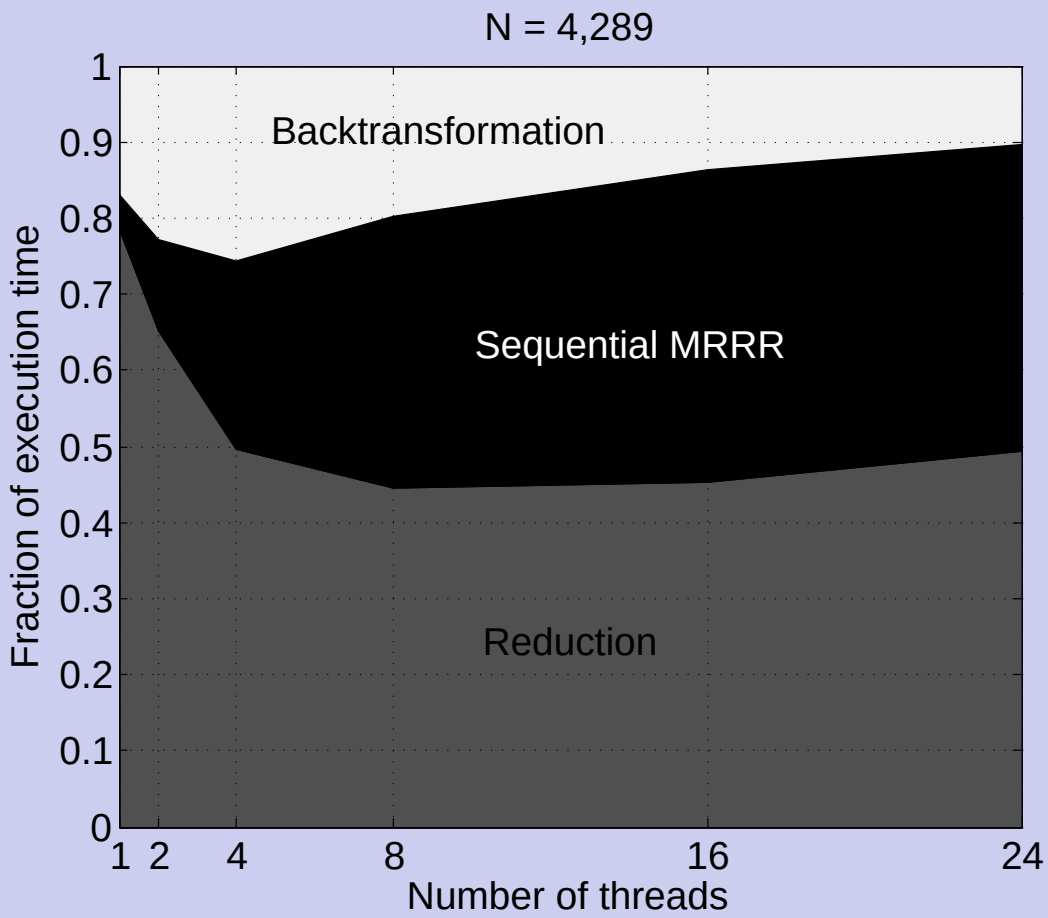
Tridiagonal eigensolver, matrix size=4289, from DFT.

More motivation?

. . . it's $O(n^2)$ anyway

More motivation?

. . . it's $O(n^2)$ anyway



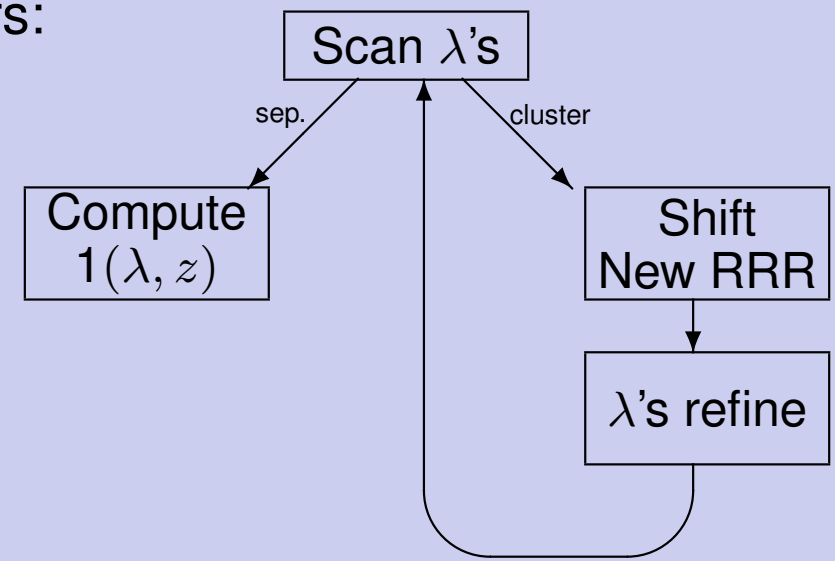
If not properly parallelized, even $O(n^2)$ dominates!

Multiple Relatively Robust Representations

- first stable algorithm to compute k eigenpairs in $O(nk)$ ops
- no reorthogonalization

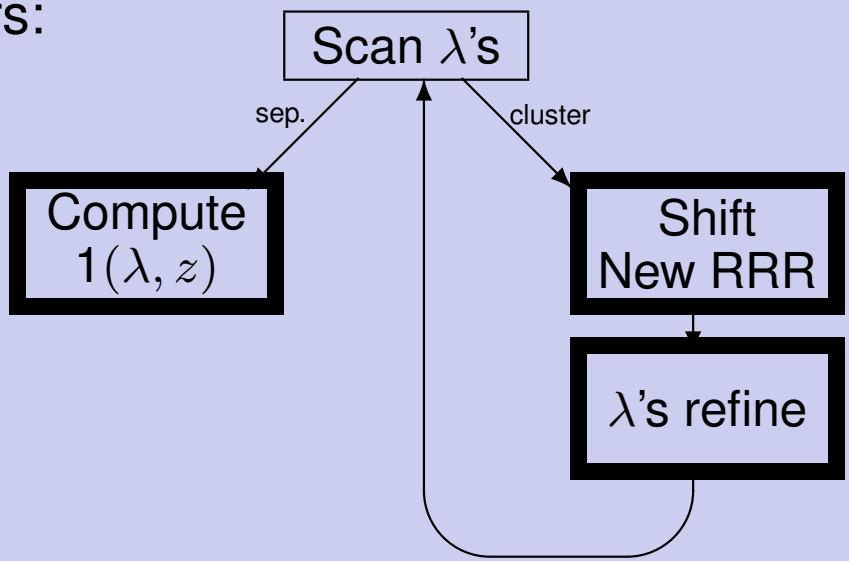
Multiple Relatively Robust Representations

- first stable algorithm to compute k eigenpairs in $O(nk)$ ops
- no reorthogonalization
- 1) eigenvalues $\rightarrow$ 2) eigenvectors + eigenvalues
- eigenvalues: *Bisection* or *dqds*
- eigenvectors:

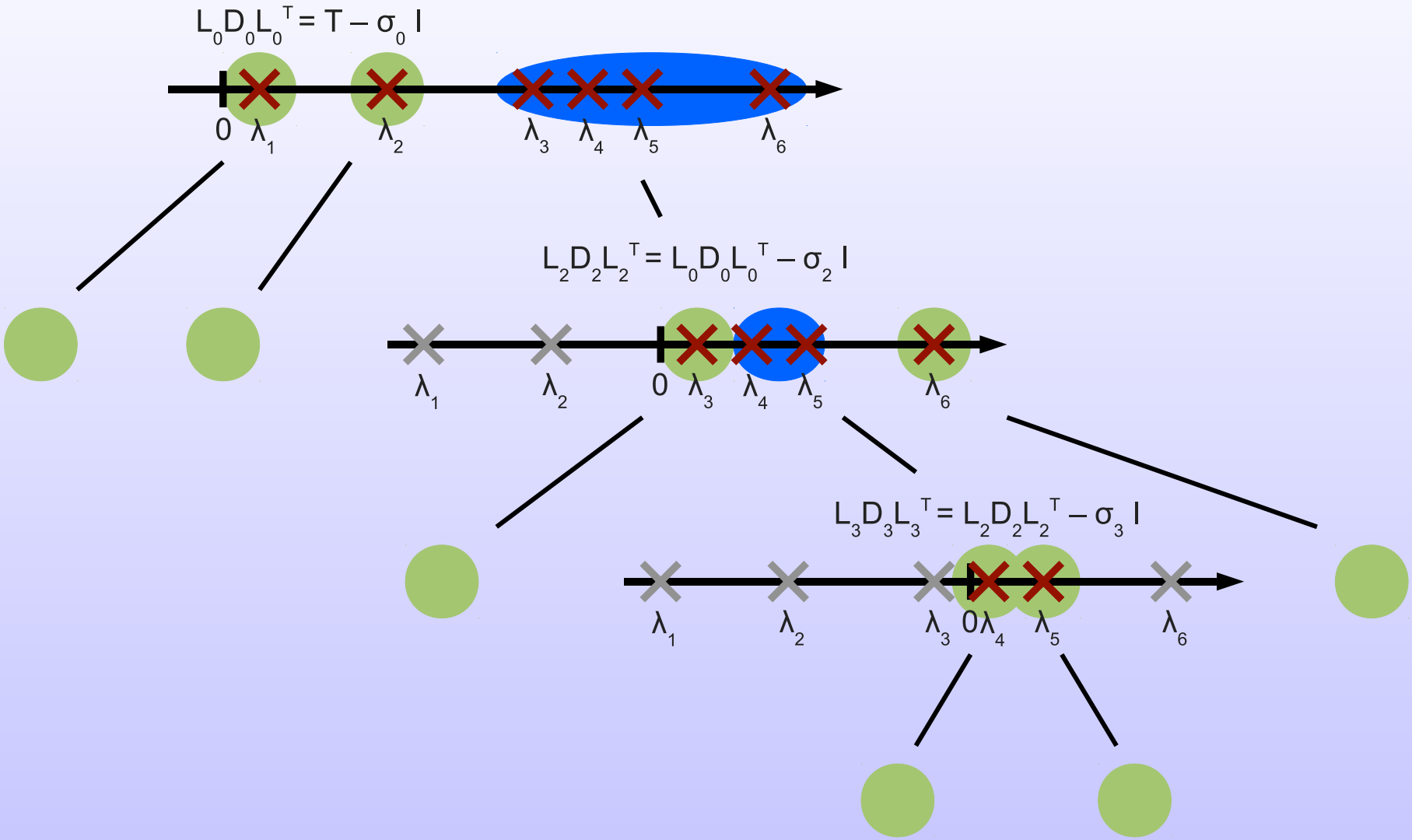


Multiple Relatively Robust Representations

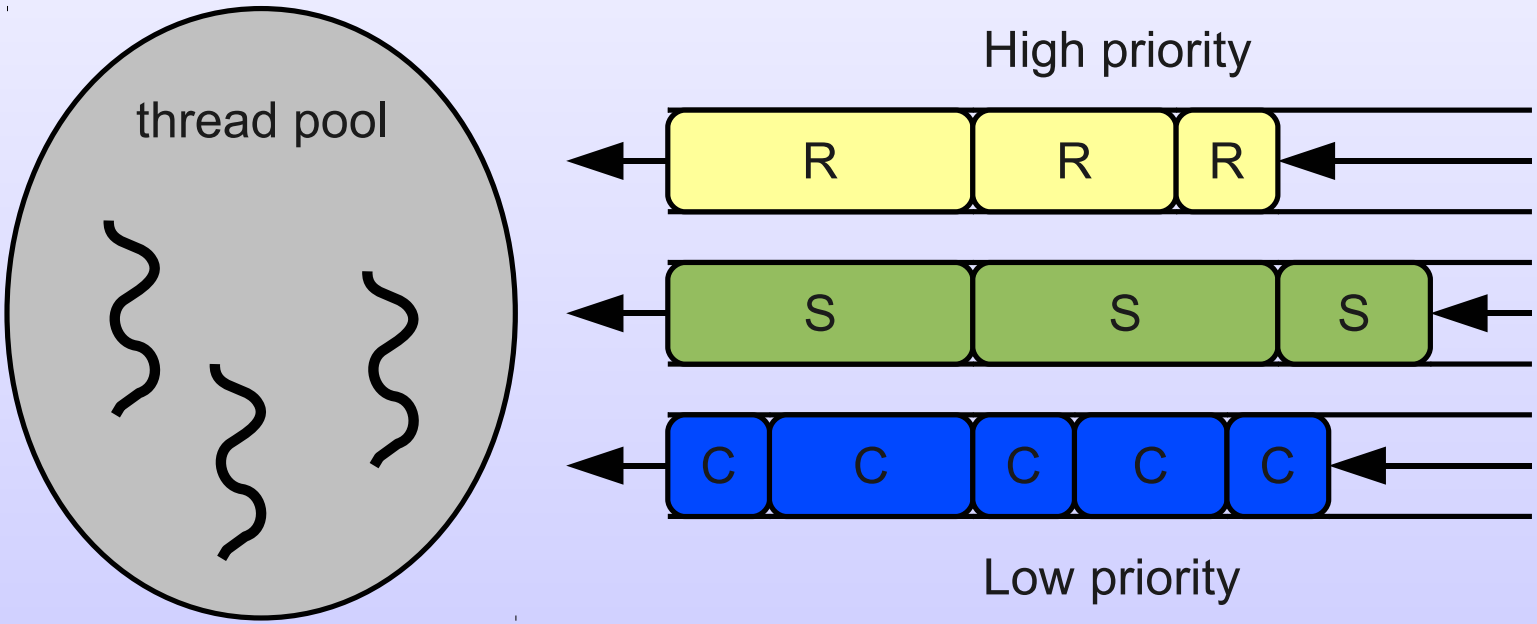
- first stable algorithm to compute k eigenpairs in $O(nk)$ ops
- no reorthogonalization
- 1) eigenvalues $\rightarrow$ 2) eigenvectors + eigenvalues
- eigenvalues: *Bisection* or *dqds*
- eigenvectors:



Representation Tree



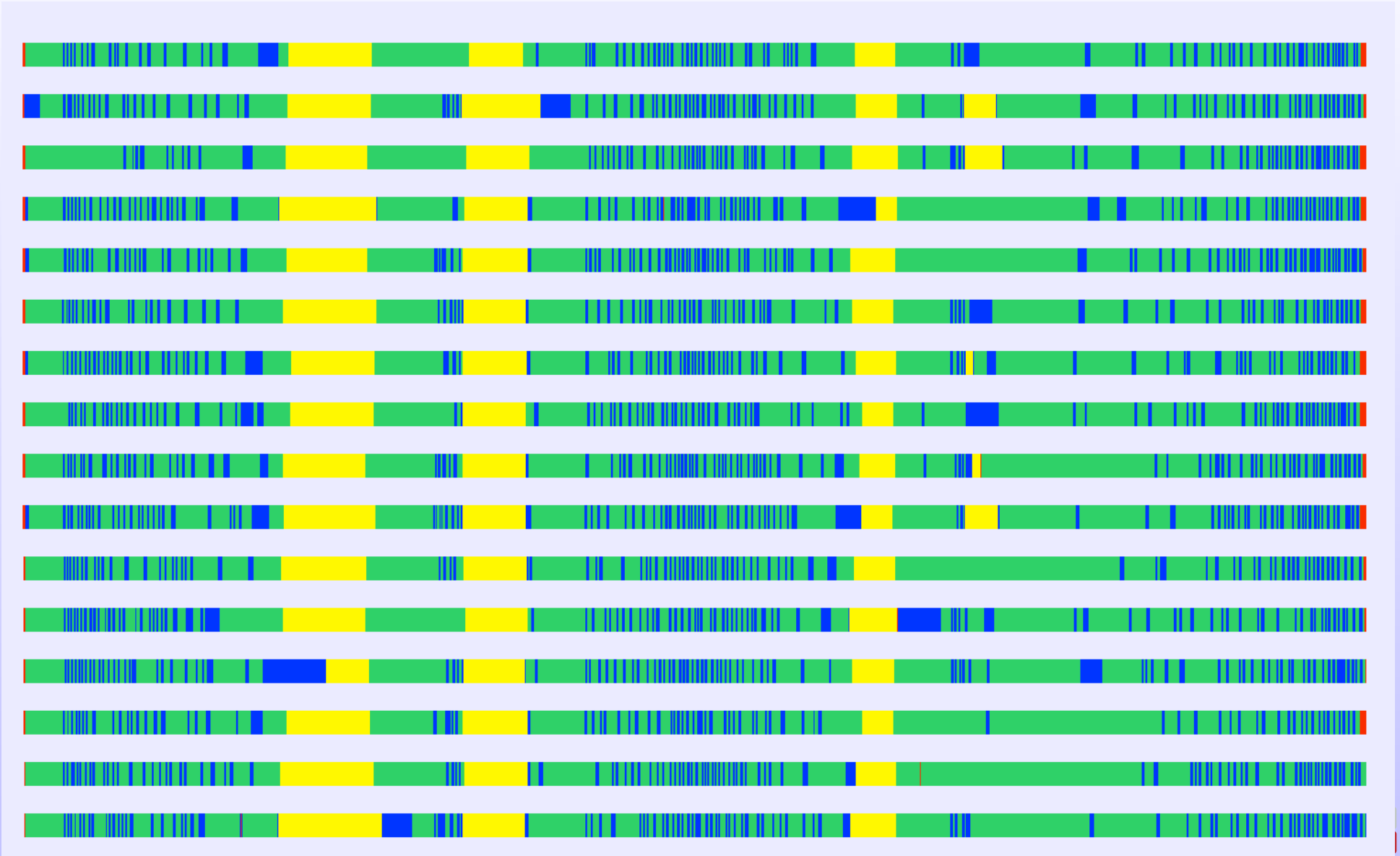
The work queue



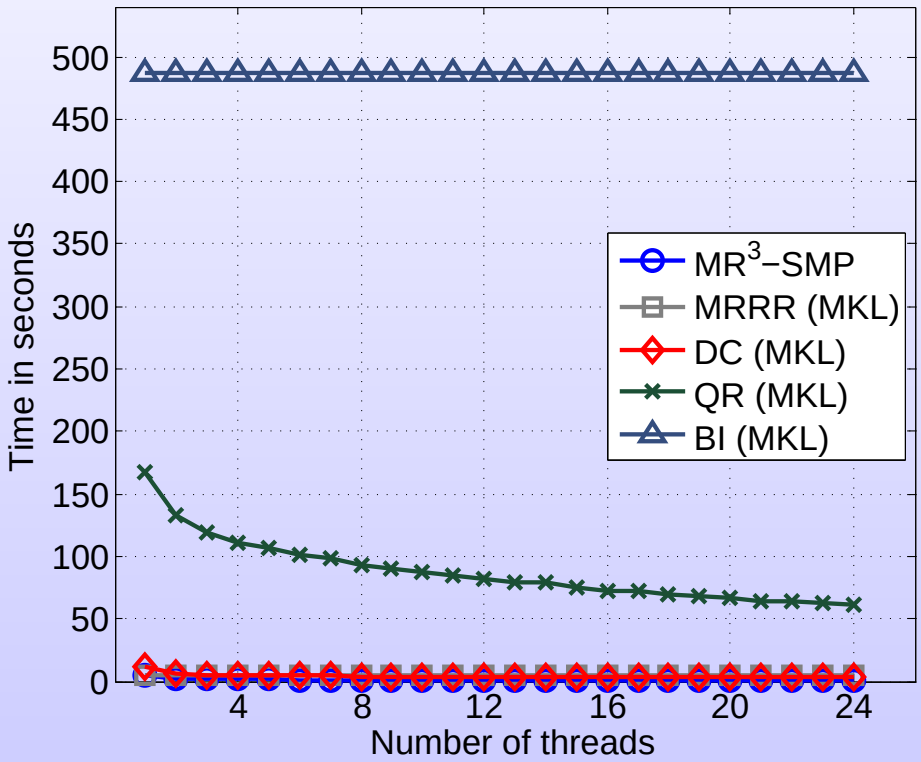
Example trace: 16 cores—eigenvectors

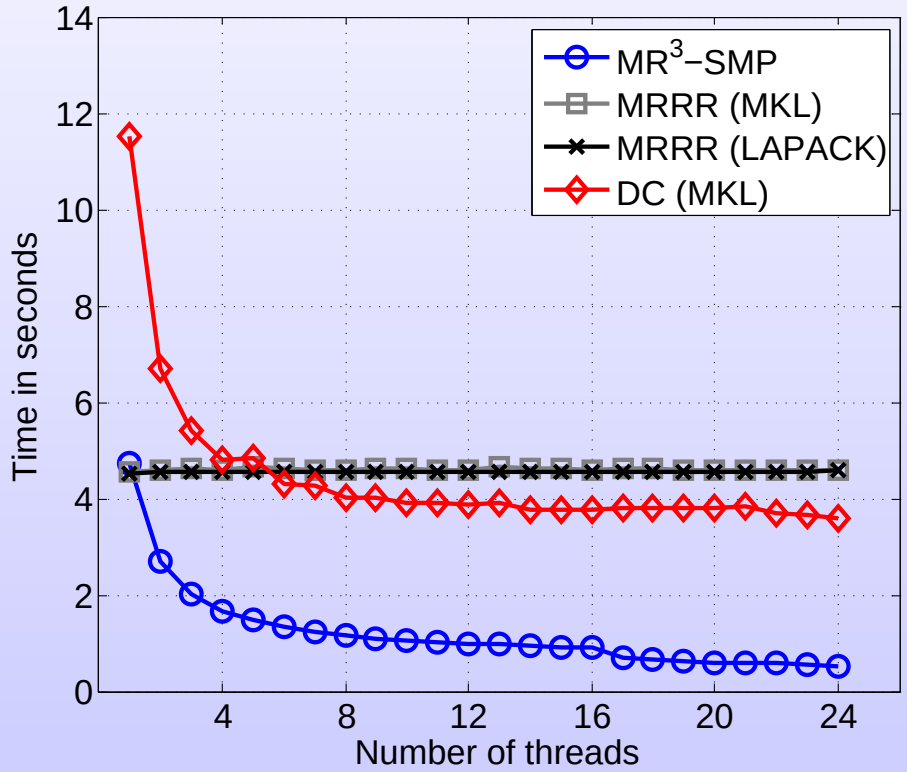
RWTH AACHEN UNIVERSITY

Matrix size: 12387 Execution time: 3.3s Sequential: 49.3s (LAPACK)



MR3-SMP: execution time

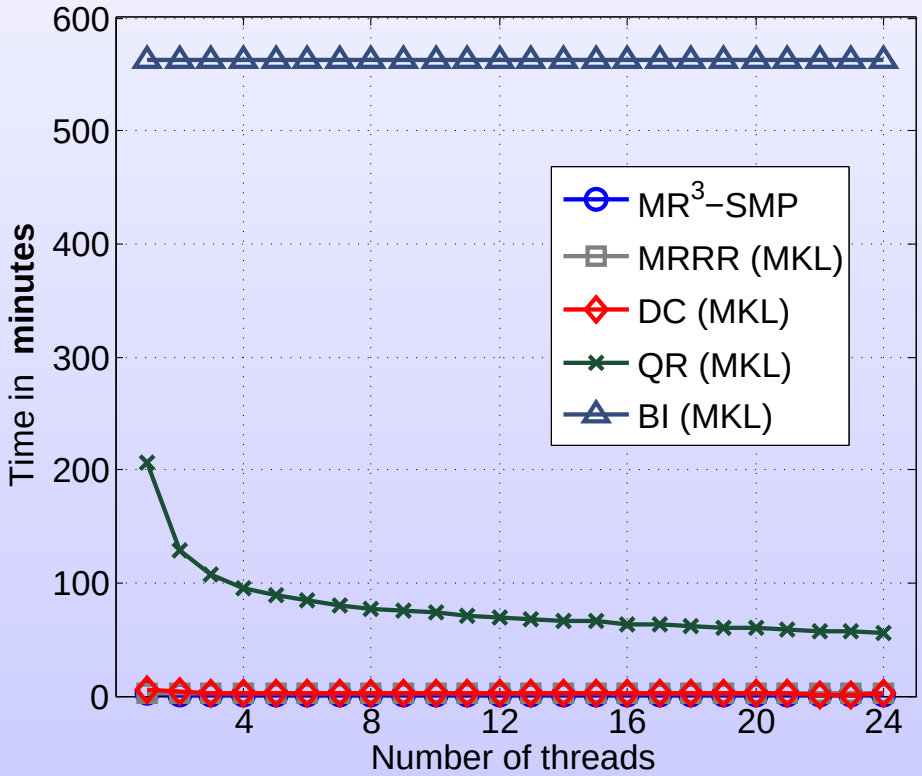


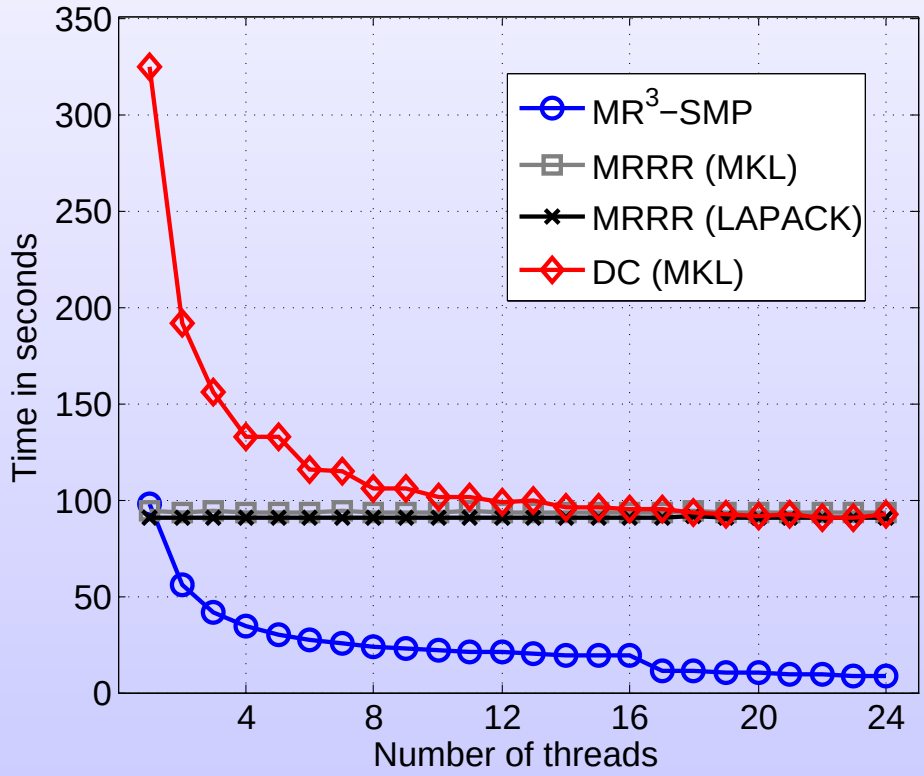


Matrix size: 4289.

A larger example: look at the scale!

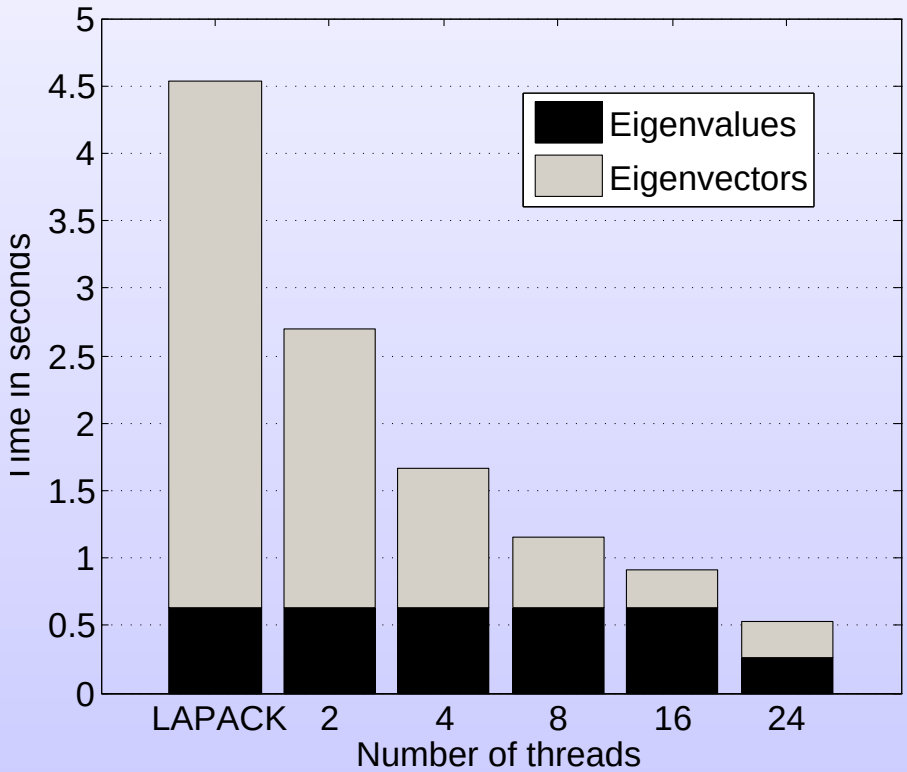
Matrix size: 16023. Frequency response analysis of automobiles.

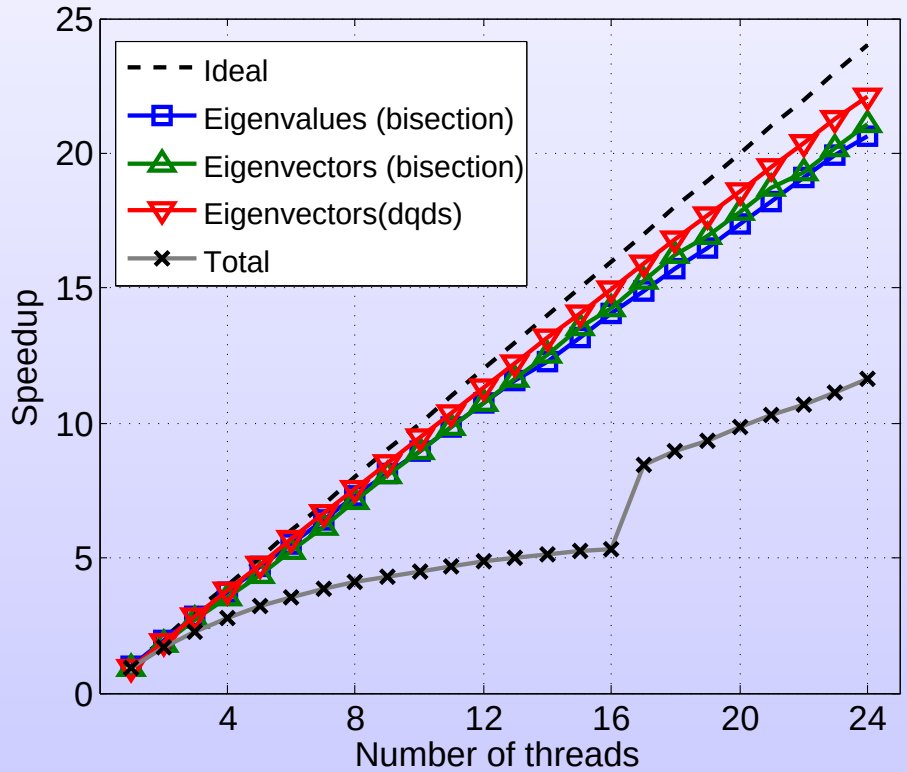




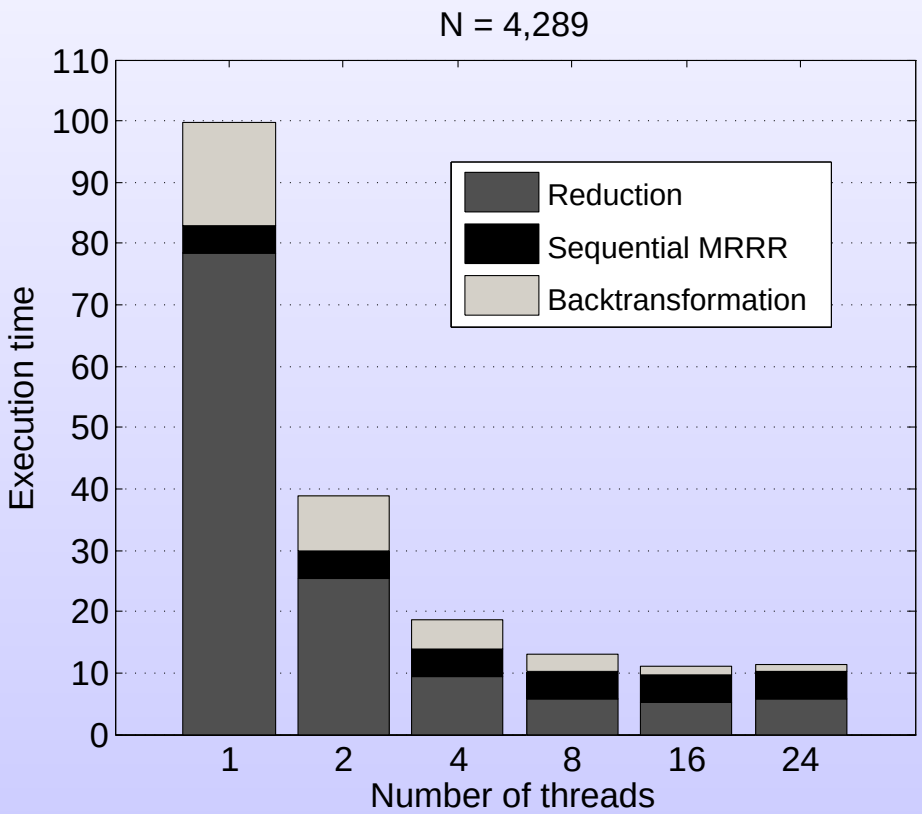
From almost 10 hours to 8.3 seconds.

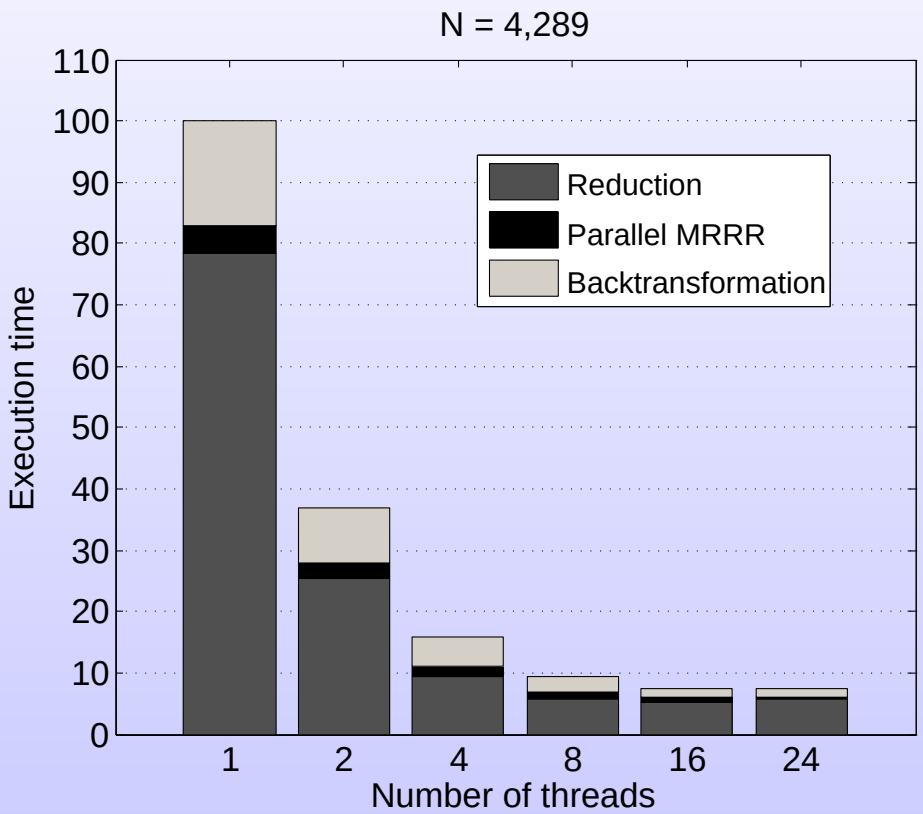
MR3-SMP: speedup



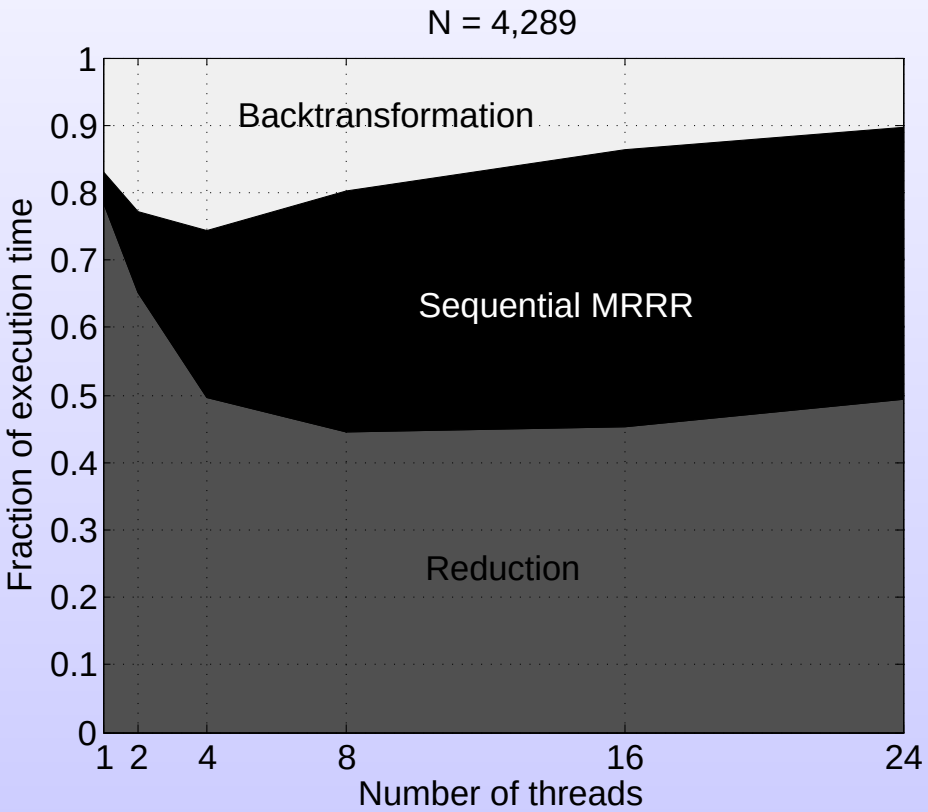


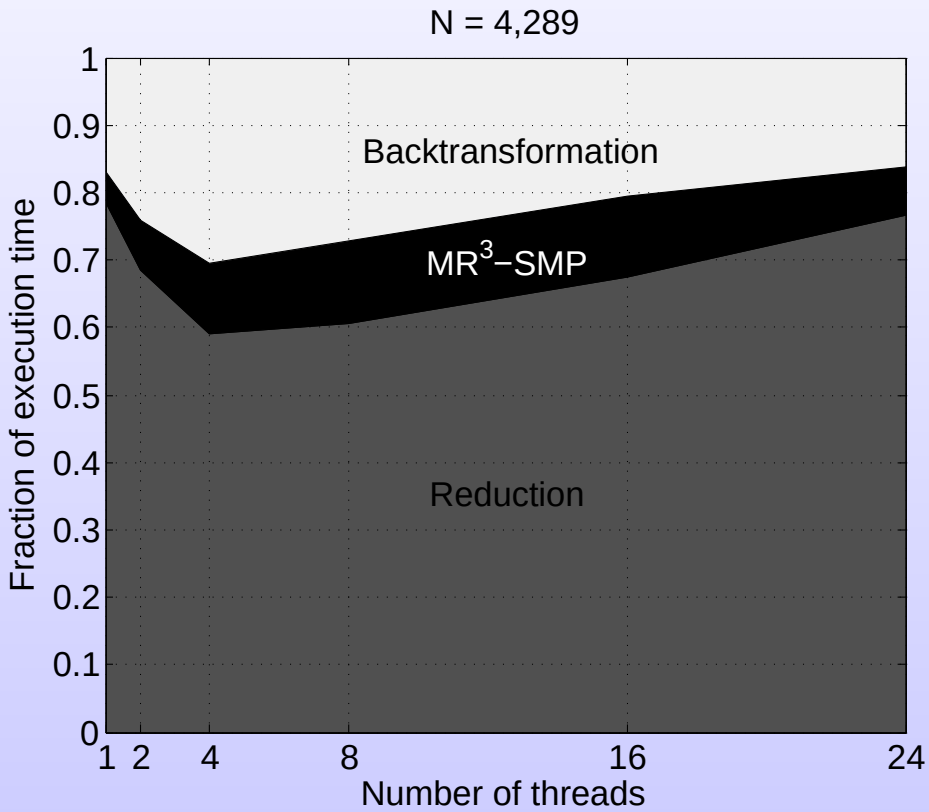
3 stages: before and after





3 stages: before and after





Conclusions

MRRR-SMP

- Matthias Petschow (AICES)
- Eigensolver tailored for multi-cores
- Almost perfect speedups
- Routines are available

- Multi-threaded BLAS for solvers: nice and easy.
- Multi-threaded BLAS for eigensolvers: not THAT good.

Conclusions

MRRR-SMP

- Matthias Petschow (AICES)
- Eigensolver tailored for multi-cores
- Almost perfect speedups
- Routines are available

- Multi-threaded BLAS for solvers: nice and easy.
- Multi-threaded BLAS for eigensolvers: not THAT good.

Thank you for the attention.

Deutsche Forschungsgemeinschaft DFG

Financial support from the Deutsche Forschungsgemeinschaft (German Research Association) through grant GSC 111 is gratefully acknowledged.