

Batched Generation of Block-Jacobi Preconditioners for Iterative Sparse Linear System Solvers on GPUs

Hartwig Anzt, Jack Dongarra, Goran Flegar, Enrique S. Quintana-Ortí,
Andrés E. Tomás

Problem setting

- Solve sparse linear system using an iterative Krylov method

$$Ax = b, \quad A \in \mathbb{R}^{n \times n}$$

Problem setting

- **Solve sparse linear system using an iterative Krylov method**

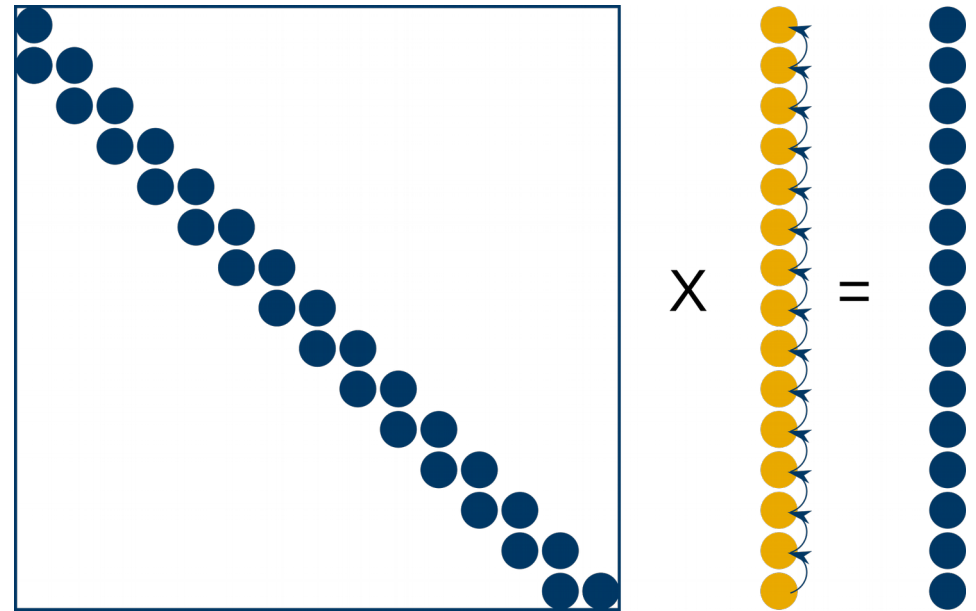
$$Ax = b, \quad A \in \mathbb{R}^{n \times n}$$

- **Convergence typically benefits from using a preconditioner**

$$M^{-1}Ax = M^{-1}b$$

Problem setting

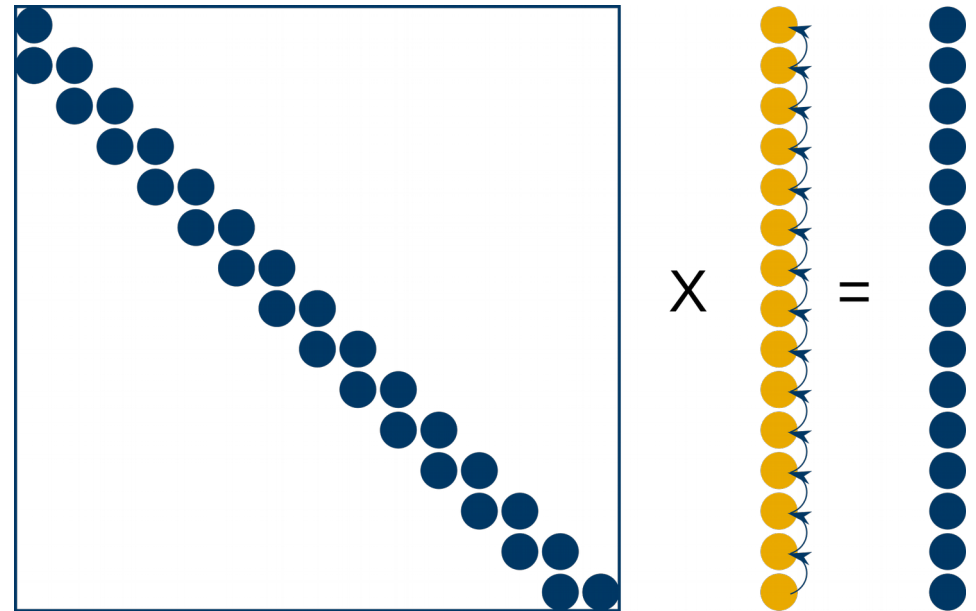
- Solve sparse linear system using an iterative Krylov method
- Convergence typically benefits from using a preconditioner
- Conventional ILU-type preconditioners offer **limited concurrency**



source: devblogs.nvidia.com/paralleforall/

Problem setting

- Solve sparse linear system using an iterative Krylov method
- Convergence typically benefits from using a preconditioner
- Conventional ILU-type preconditioners offer **limited concurrency**
- **Need high degree of parallelism** to use a GPU effectively
 - ILU on GPUs is an area of active research



source: devblogs.nvidia.com/parallelforall/

GPU architecture (NVIDIA GP100)

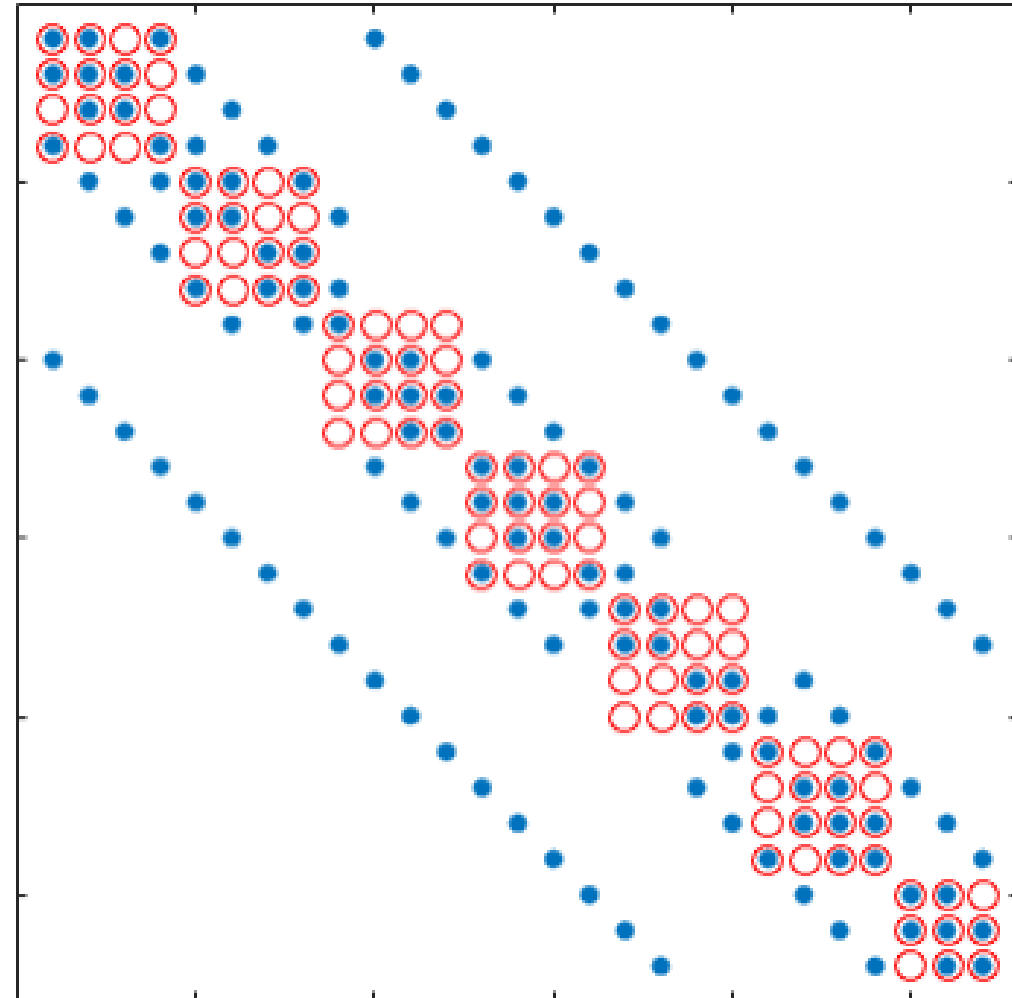
- **56 SMs x 64 cores = 3584 cores!**
- **Need to oversubscribe to hide memory latency!**



source: devblogs.nvidia.com/parallelforall/

Different approach

- **Scalar Jacobi**
 - Scale with inverses of diagonal elements
- **Block-Jacobi**
 - Scale with inverses of diagonal blocks (possibly of different sizes!)
 - Can reflect the block structure of the problem
 - Often superior to scalar Jacobi
 - Lower convergence rate improvement compared to ILU
- **Can process each block independently!**



- **Restrict block size to 32x32**
 - Large block sizes require more memory to store the preconditioner matrix

- **Restrict block size to 32x32**
 - Large block sizes require more memory to store the preconditioner matrix
- **Use a single warp to process the whole block (one thread per column)**
 - No need for explicit synchronization



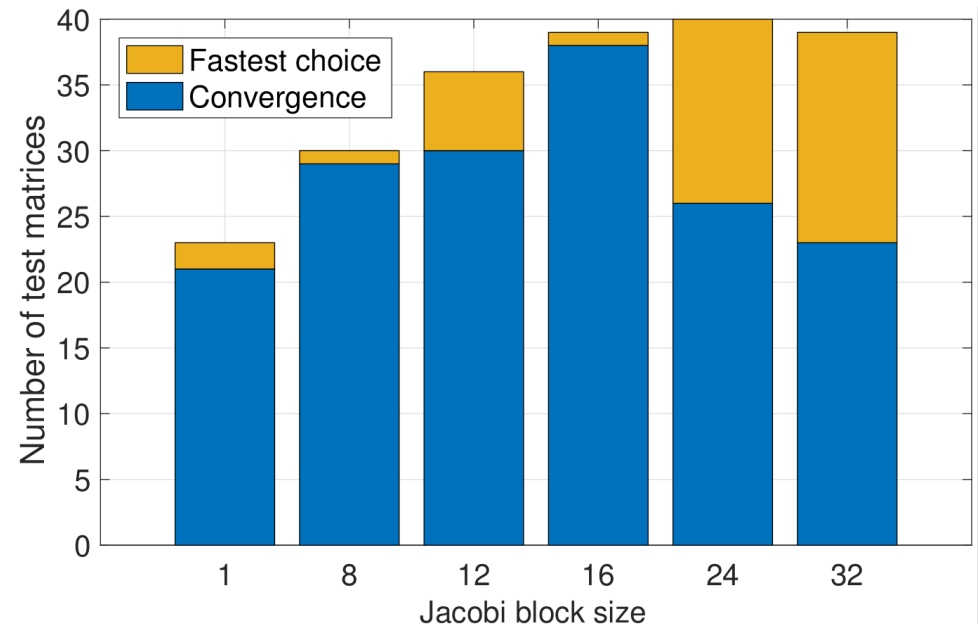
General Ideas

- **Restrict block size to 32x32**
 - Large block sizes require more memory to store the preconditioner matrix
- **Use a single warp to process the whole block (one thread per column)**
 - No need for explicit synchronization
- **Use the large register file to store the entire block**
 - Read/write from mem. once
 - Avoids load/store instructions
 - Com. via warp shuffles



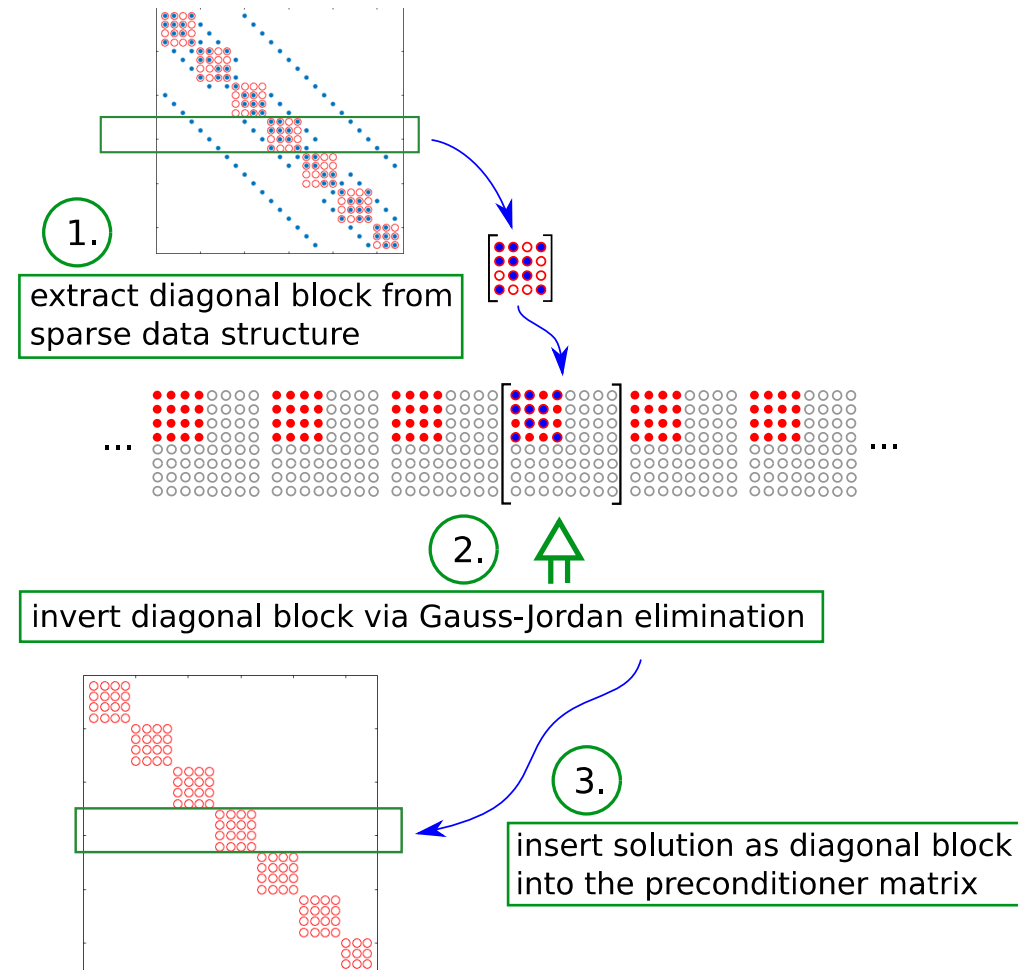
Benefits of block-Jacobi

- 40 matrices from SuiteSparse
- MAGMA-sparse open source library
 - IDR solver
 - Jacobi preconditioner
 - Supervariable blocking
- **Block-Jacobi improves the robustness of the solver**
 - More problems converge
- **Decreases time-to-solution**



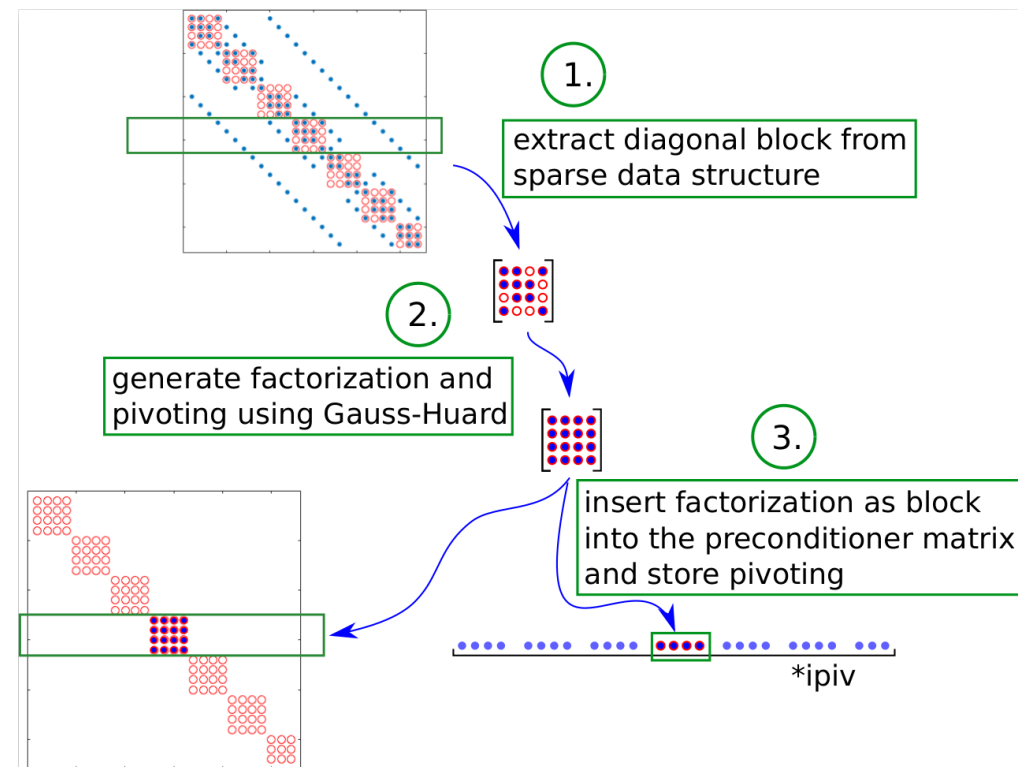
Implementation options

- Inversion in preconditioner setup + matrix-vector product in application
 - (FLOPS: $2n^3$ setup, $2n^2$ app.)
 - H. Anzt et al., “Batched Gauss-Jordan Elimination for Block-Jacobi Preconditioner Generation on GPUs”, PMAM’17

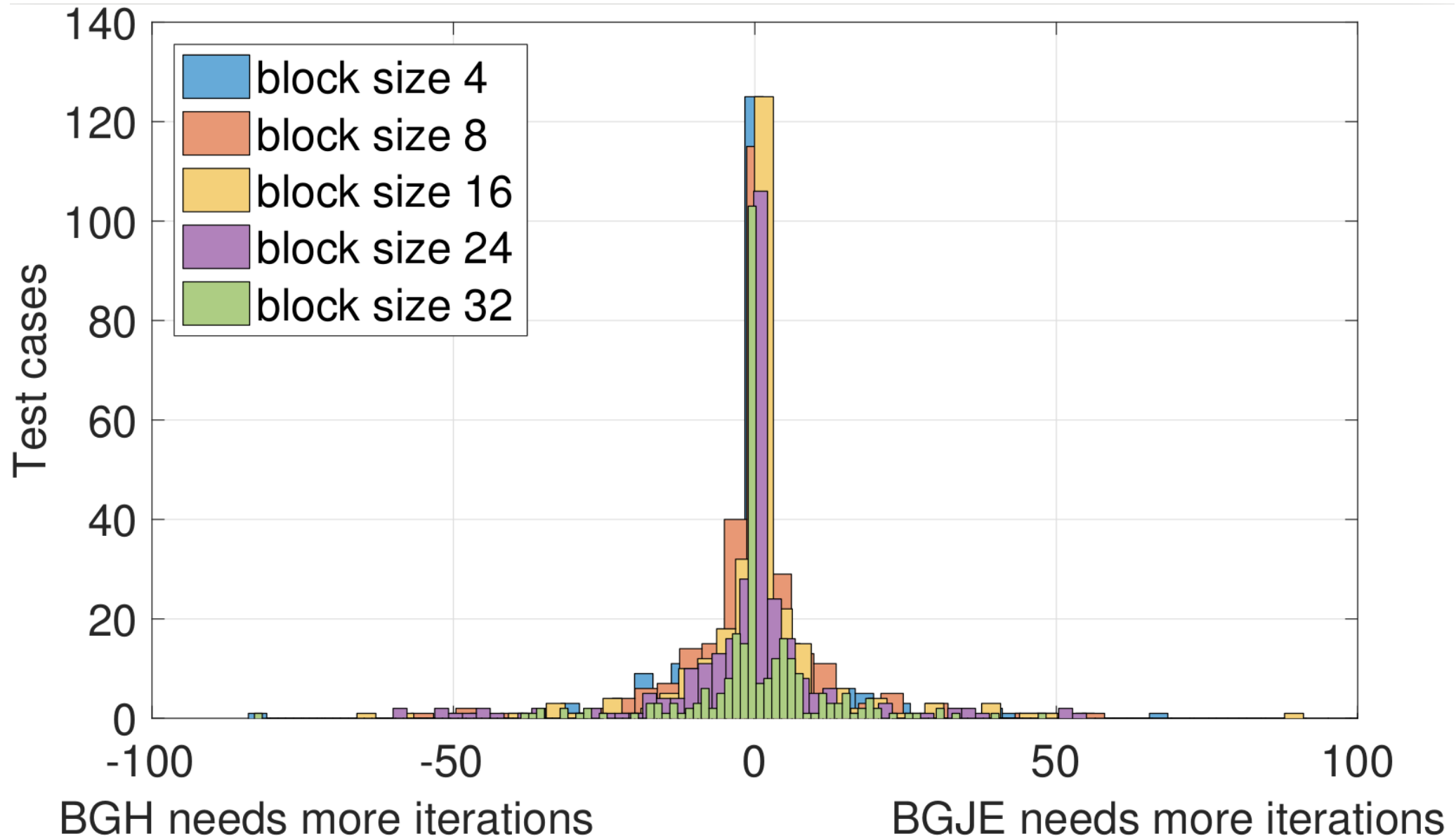


Different options

- **Inversion in preconditioner setup + matrix-vector product in application**
 - (FLOPS: $2n^3$ setup, $2n^2$ app.)
 - H. Anzt et al., “Batched Gauss-Jordan Elimination for Block-Jacobi Preconditioner Generation on GPUs”, PMAM’17
- **Matrix decomposition in setup + solve in application**
 - (FLOPS: $2/3n^3$ setup, $2n^2$ app.)
 - Gauss-Huard decomposition



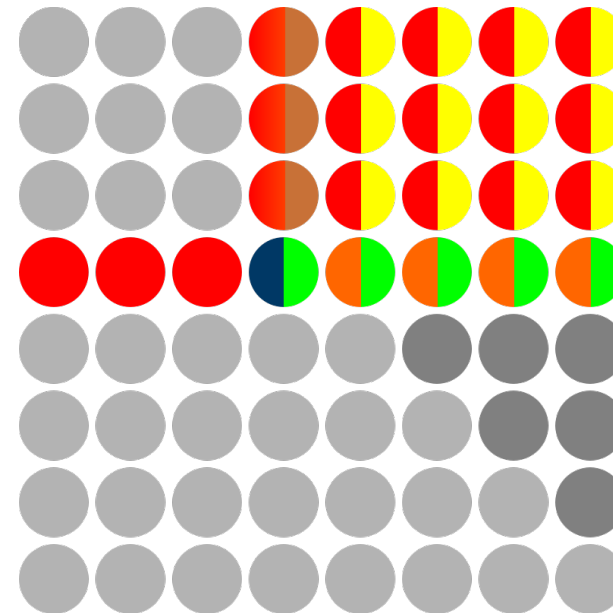
Inversion?!



Gauss-Huard decomposition

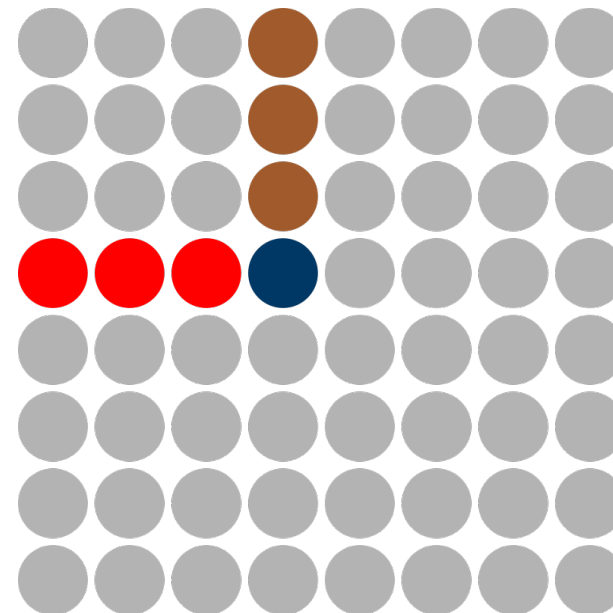
- **Decomposition**

- GEMV ($G = G - RR$)
- SCAL ($O = O / B$)
- GER ($Y = Y - BO$)
- Column pivoting
 - Do not swap the columns, just remember which thread holds which column of the result



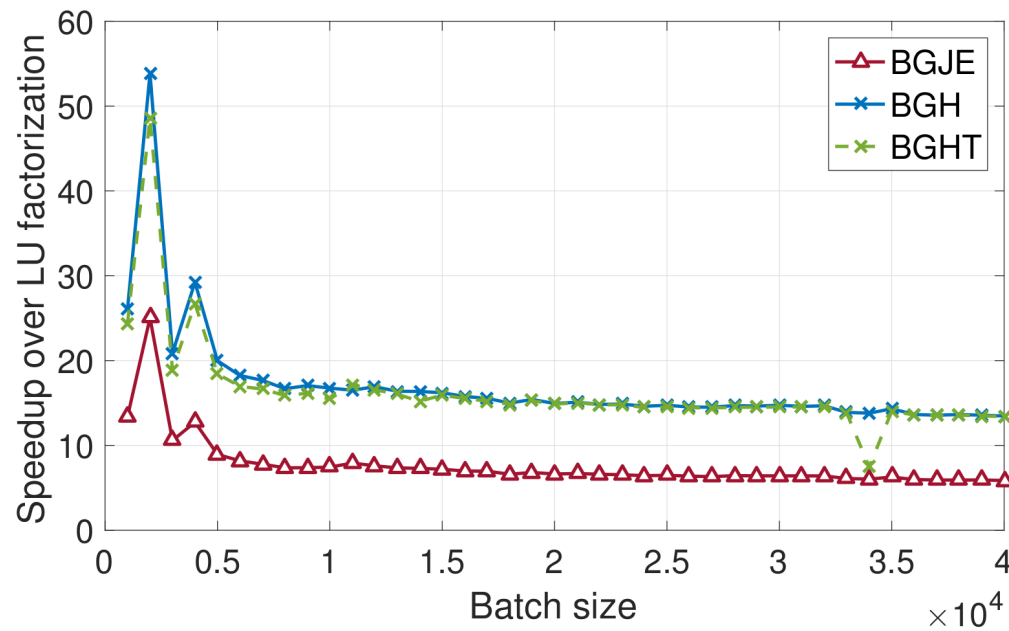
- **Solve**

- Store only the solution vector into registers
- DOT ($G = G - RR$)
- SCAL ($O = O / B$)
- AXPY ($Y = Y - BO$)
- Store lower part transposed wrp. to anti-diagonal for coalesced mem. access

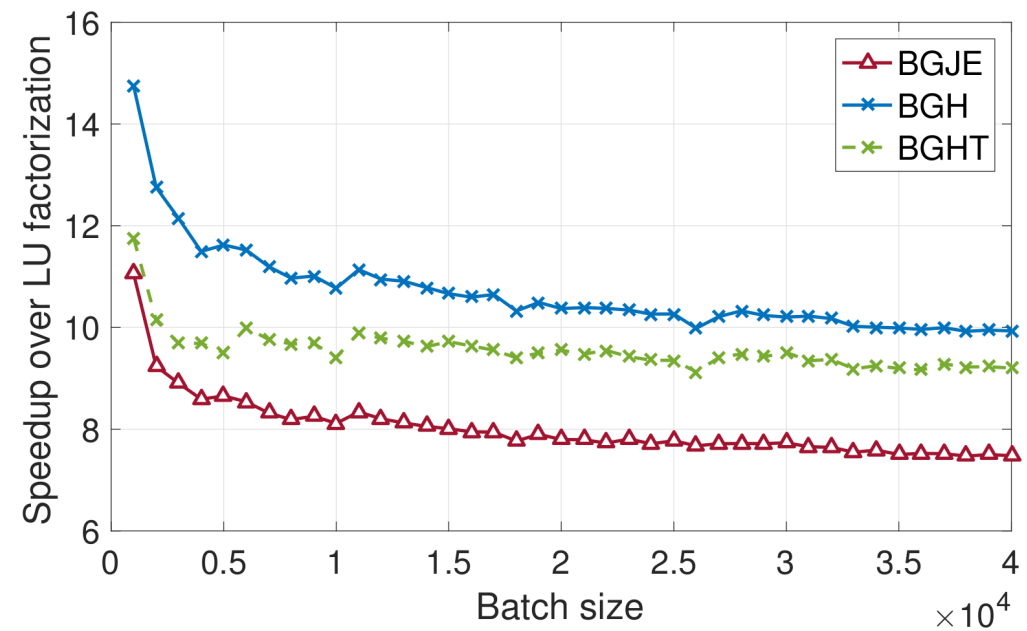


Comparison to batched LU (MAGMA)

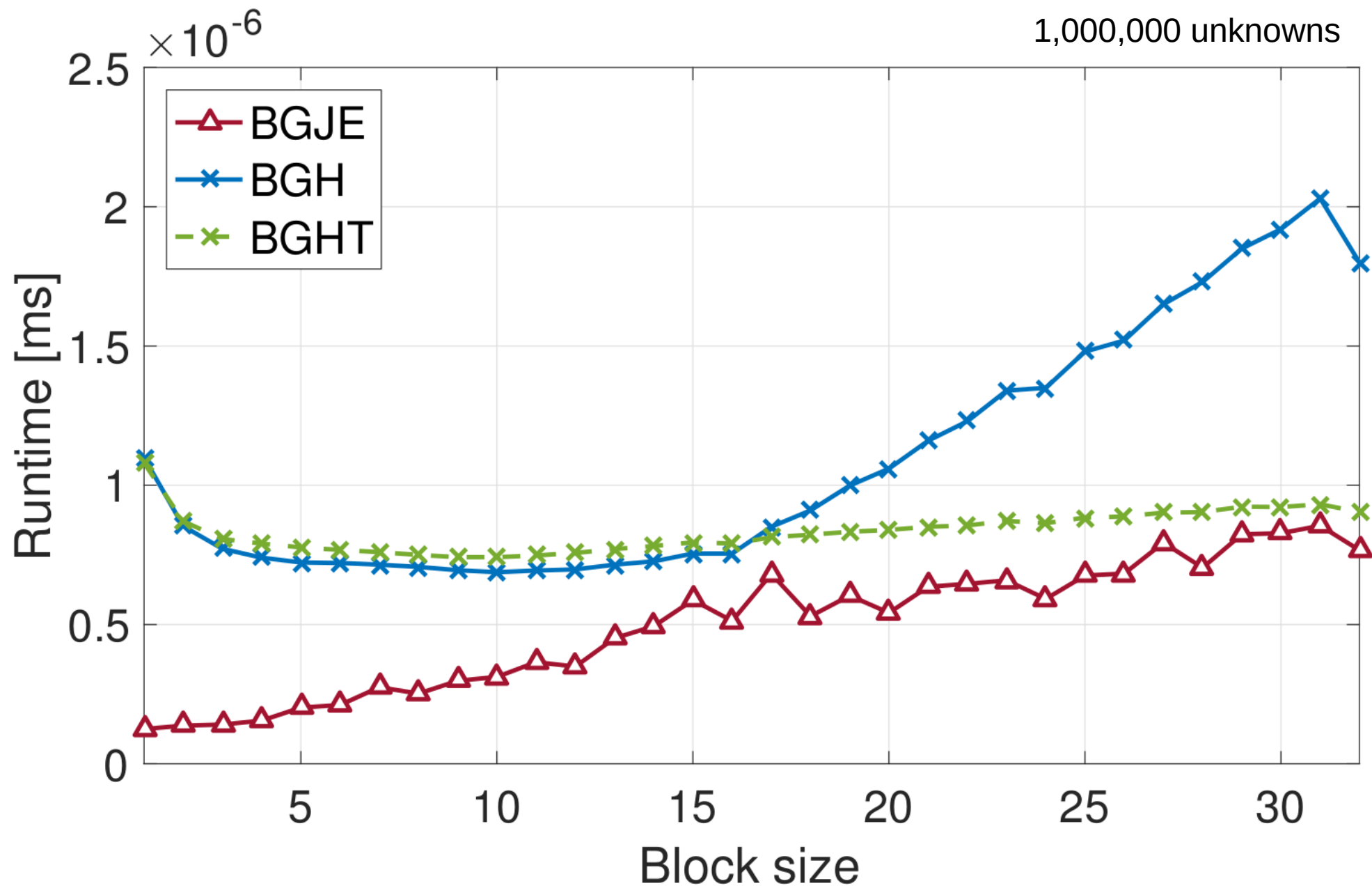
Block size 16



Block size 32

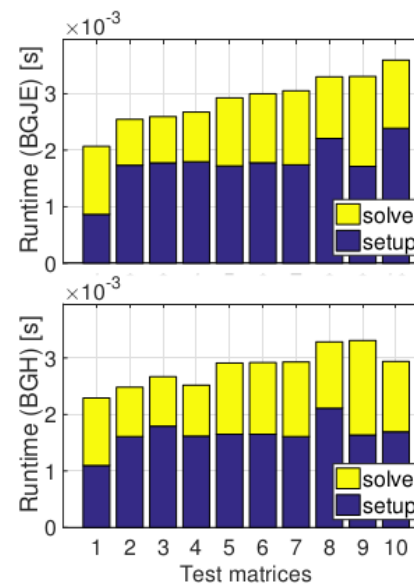
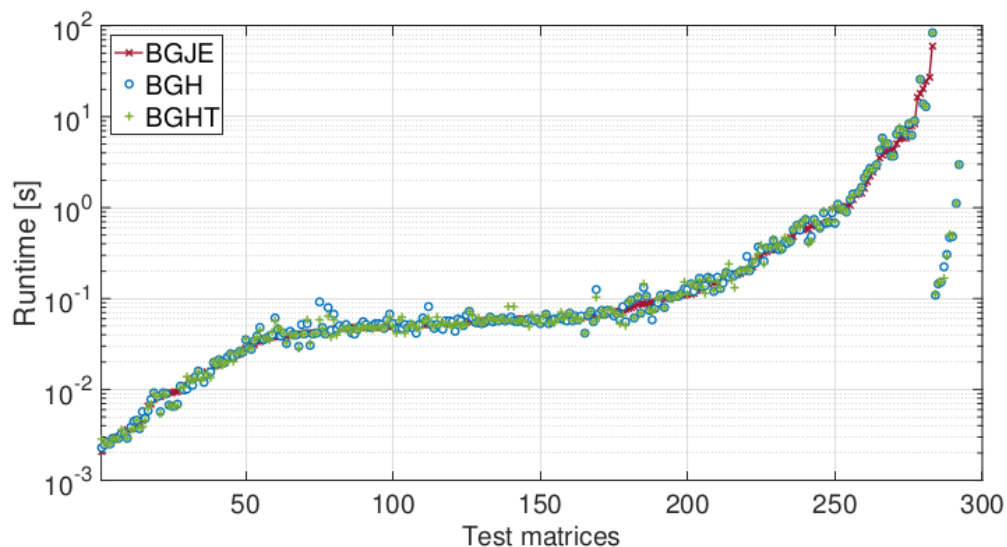


Application time

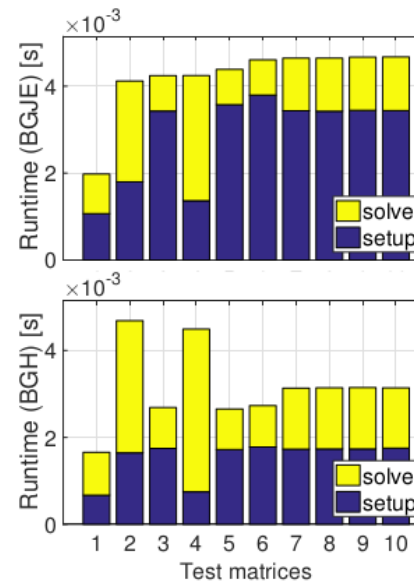
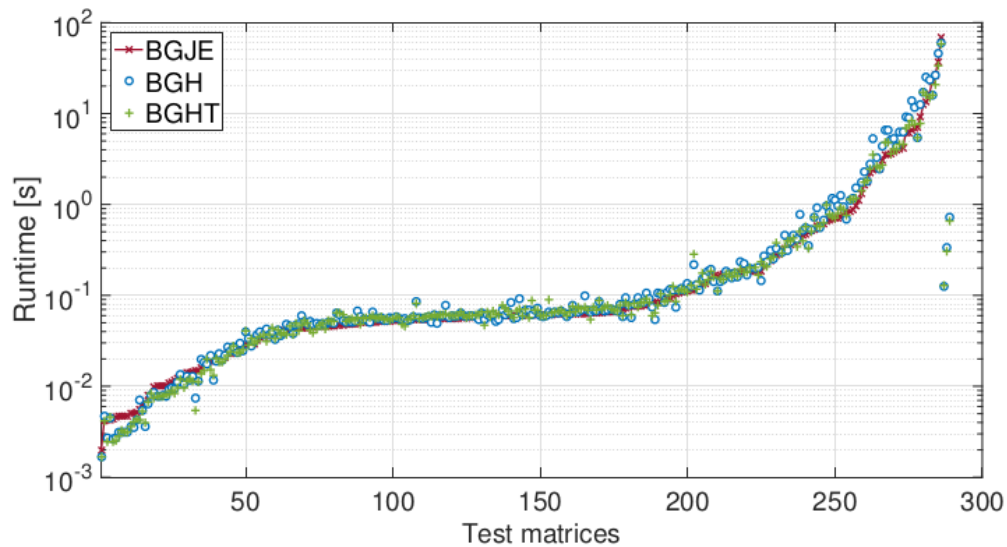


Total runtime of block-Jacobi preconditioned BiCGSTAB

Block size 16



Block size 32



Thank you! Questions?

All functionalities are part of the MAGMA-sparse project.

MAGMA SPARSE

ROUTINES	BiCG, BiCGSTAB, Block-Asynchronous Jacobi, CG, CGS, GMRES, IDR, Iterative refinement, LOBPCG, LSQR, QMR, TFQMR
-----------------	--

PRECONDITIONERS	ILU / IC, Jacobi, ParILU, ParILUT, Block Jacobi, ISAI
------------------------	---

KERNELS	SpMV, SpMM
----------------	------------

DATA FORMATS	CSR, ELL, SELL-P, CSR5, HYB
---------------------	-----------------------------

This research is based on a cooperation between Hartwig Anzt, Jack Dongarra (University of Tennessee), Goran Flegar, Enrique S. Quintana-Ortí and Adrés E. Tomás (Universidad Jaume I)

