Adaptive-Precision Preconditioning for Sparse Linear Systems

Hartwig Anzt, Jack Dongarra**, Goran Flegar**, Nick Higham, **Enrique S. Quintana-Ortí**

Motivation: where is the energy spent?

Operation	approximate energy cost
DP floating point multiply-add	100 pJ
DP DRAM read-to-register	4800 pJ
DP word transmit-to-neighbor	7500 pJ
DP word transmit-across-system	9000 pJ

48x

*John Shalf (LBNL)

Motivation: where is the energy spent?

Which are the communication-intensive algorithms?

- Sorting
- Graph Algorithms
- Sparse Linear Algebra
  - Sparse Direct Solvers *(1D problems, 2D problems)**
  - Sparse Iterative Solvers *(2D problems on manycore, 3D problems)**
    - Multigrid , Krylov Methods …
    - Preconditioning: Incomplete LU, (block-) Jacobi, Gauss-Seidel...

*Mike Heroux (SNL)

Block-Jacobi Preconditioning

- **Jacobi method** based on **diagonal scaling**: $P = diag(A)$
  - Can be used as iterative solver:

$$x^{(k+1)} = x^{(k)} + P^{-1}b - P^{-1}Ax^{(k)}$$

  - Can be used as preconditioner: $\tilde{A} = P^{-1}A$, $\tilde{b} = P^{-1}b$

$$Ax = b \Leftrightarrow \tilde{A}x = \tilde{b}$$

Block-Jacobi Preconditioning

- **Jacobi method** based on **diagonal scaling**: $P = diag(A)$
  - Can be used as iterative solver:

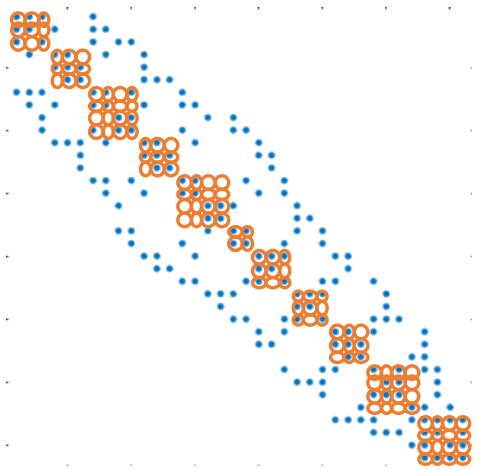
$$x^{(k+1)} = x^{(k)} + P^{-1}b - P^{-1}Ax^{(k)}$$

  - Can be used as preconditioner: $\tilde{A} = P^{-1}A$, $\tilde{b} = P^{-1}b$

$$Ax = b \Leftrightarrow \tilde{A}x = \tilde{b}$$

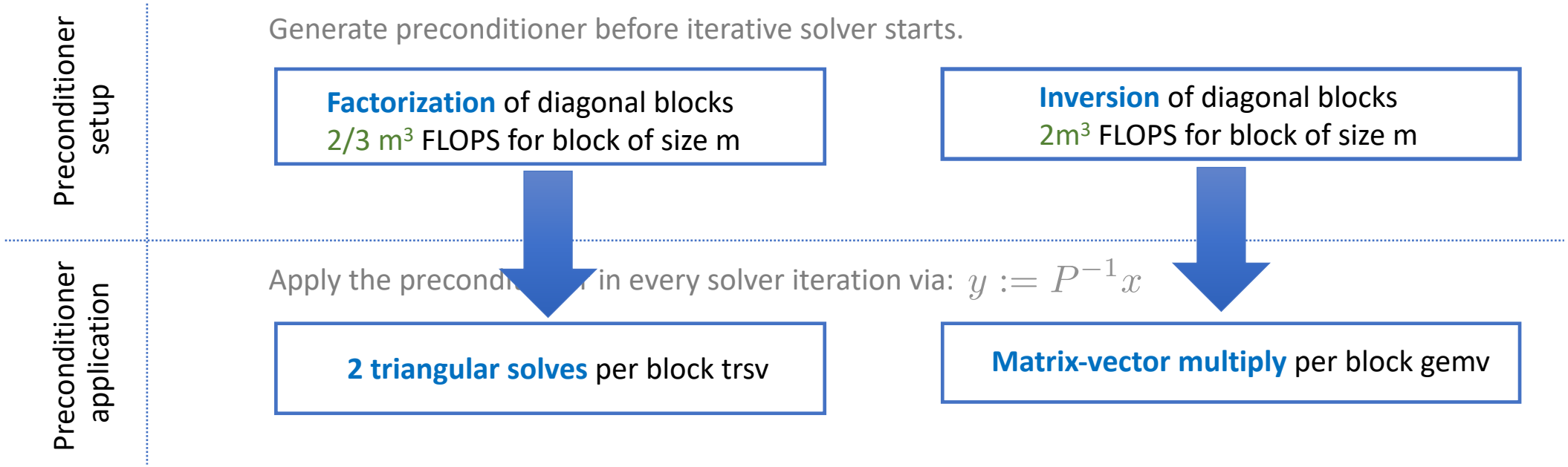
- **Block-Jacobi** is based on **block-diagonal scaling**: $P = diag_B(A)$
  - Large set of small diagonal blocks.
  - Each block corresponds to one (small) linear system.
    - *Larger* blocks typically **improve convergence**.
    - *Larger* blocks make block-Jacobi **more expensive**.

      Extreme case: one block of matrix size.

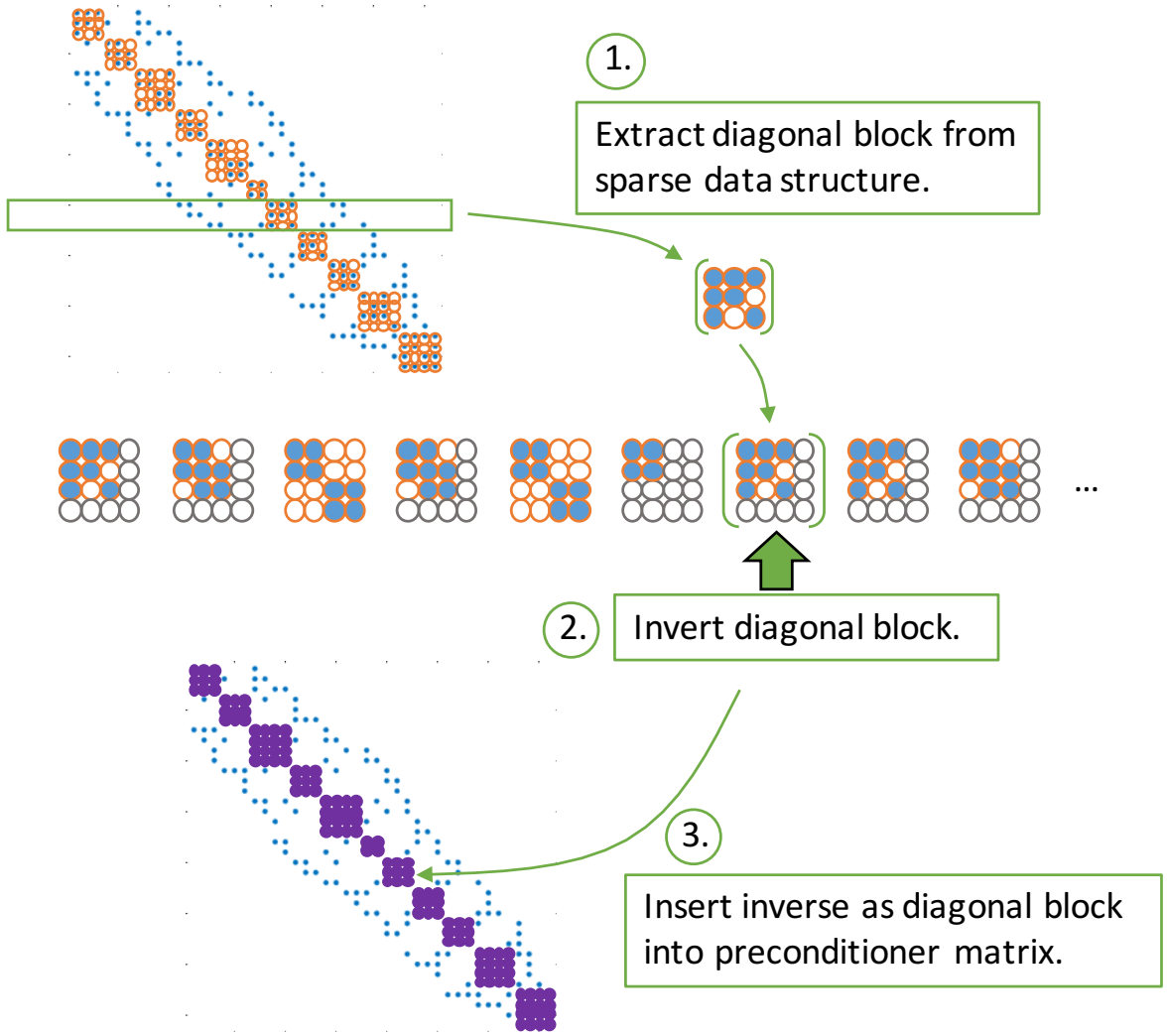


Block-Jacobi Preconditioning

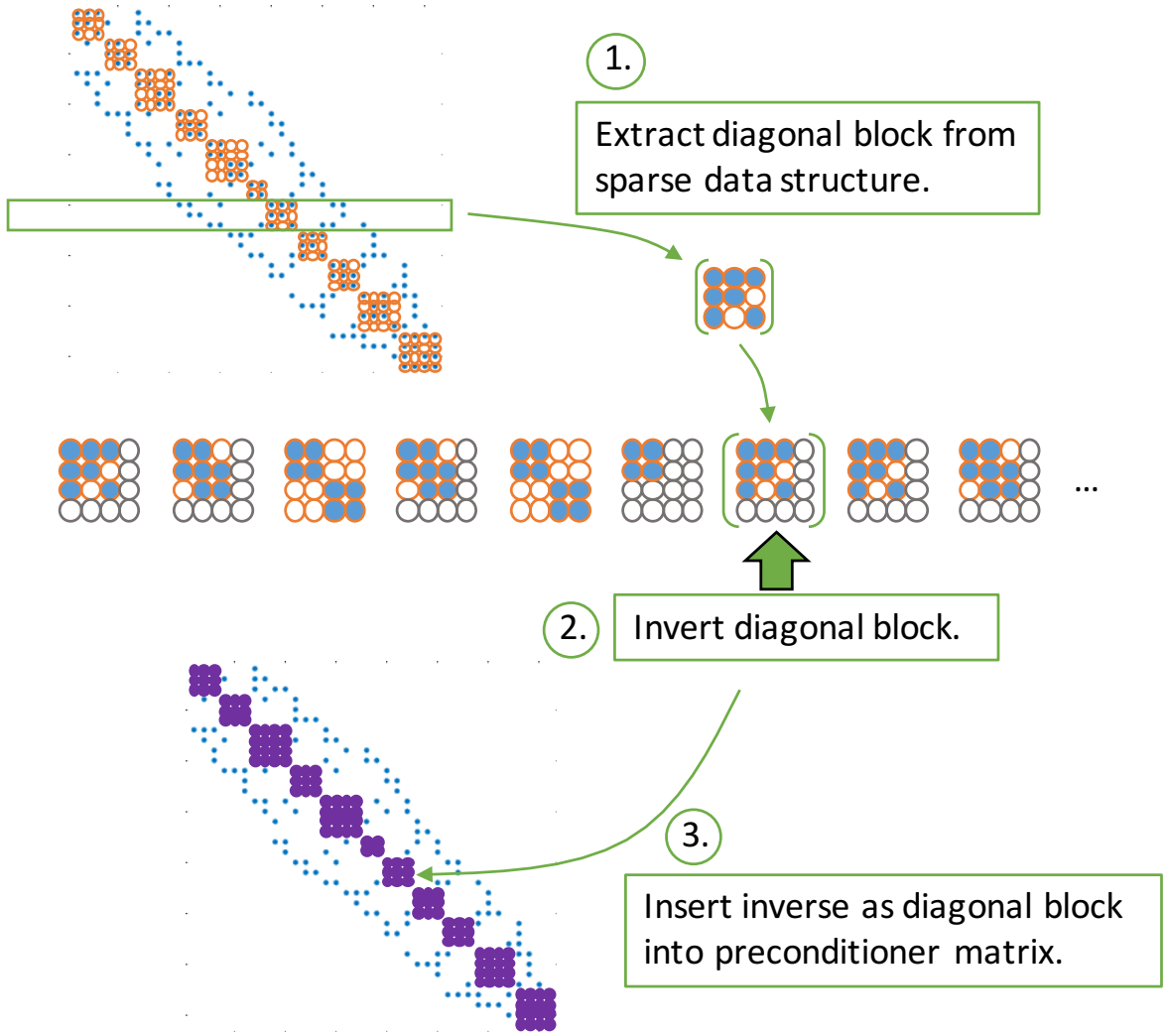
- **Block-Jacobi** method typically used as **preconditioner** inside Krylov solver.
- Target: large, sparse linear systems.
- **FEM** discretizations often carry a block-structure (multiple variables per node).
- "Natural blocks" of **small size** (8, 12,...).
- System matrix often stored in **sparse data structure** (CSR).



Inversion-based Block-Jacobi Preconditioning



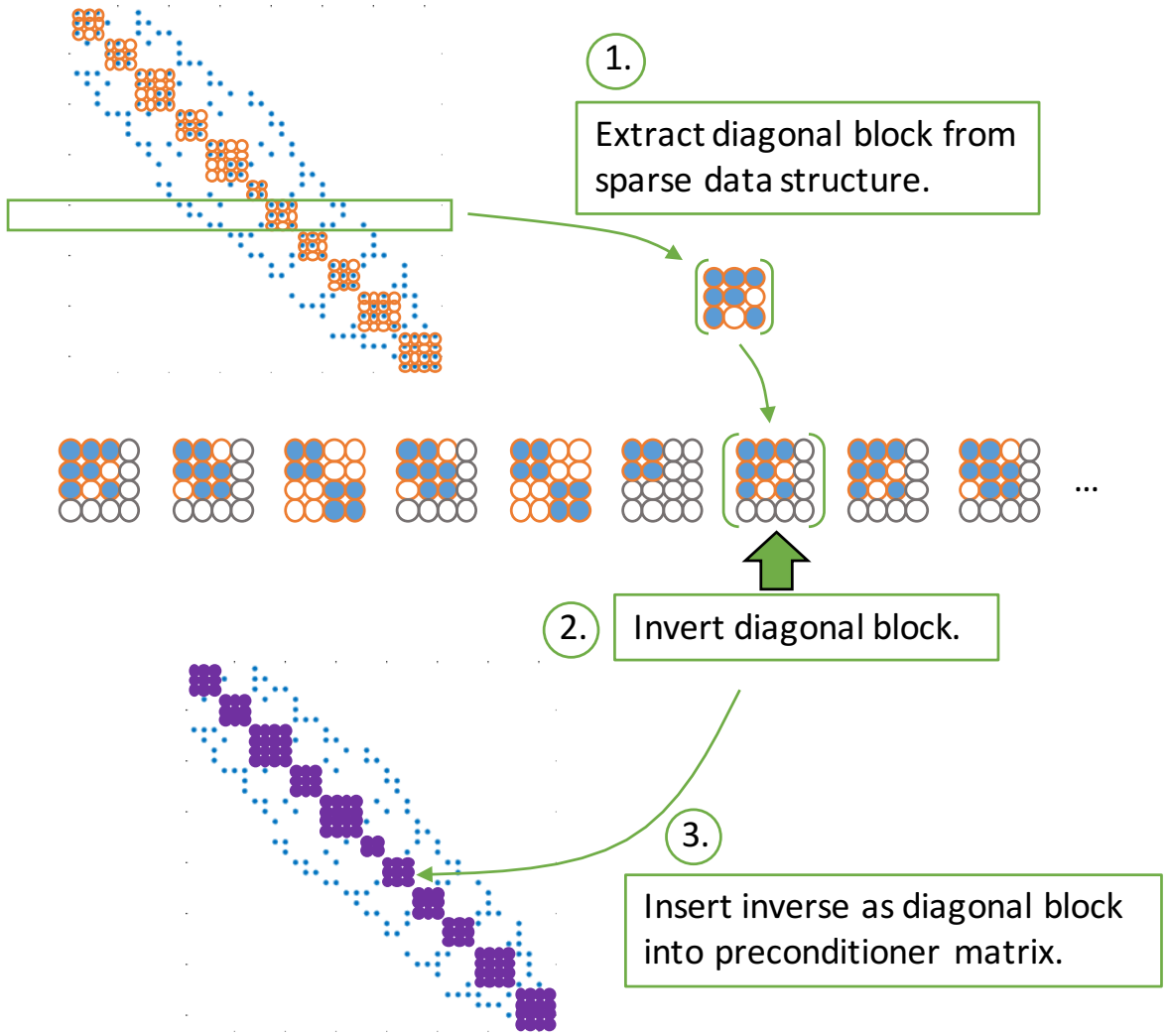
Inversion-based Block-Jacobi Preconditioning



- Cost of **Inversion**:
  $2m_i^3$ FLOPs for block of size m_i.

$$\sum_{blocks} 2m_i^3$$

Inversion-based Block-Jacobi Preconditioning



- Cost of **Inversion:**
  $2m_i^3$ FLOPs for block of size m_i.

$$\sum_{blocks} 2m_i^3$$

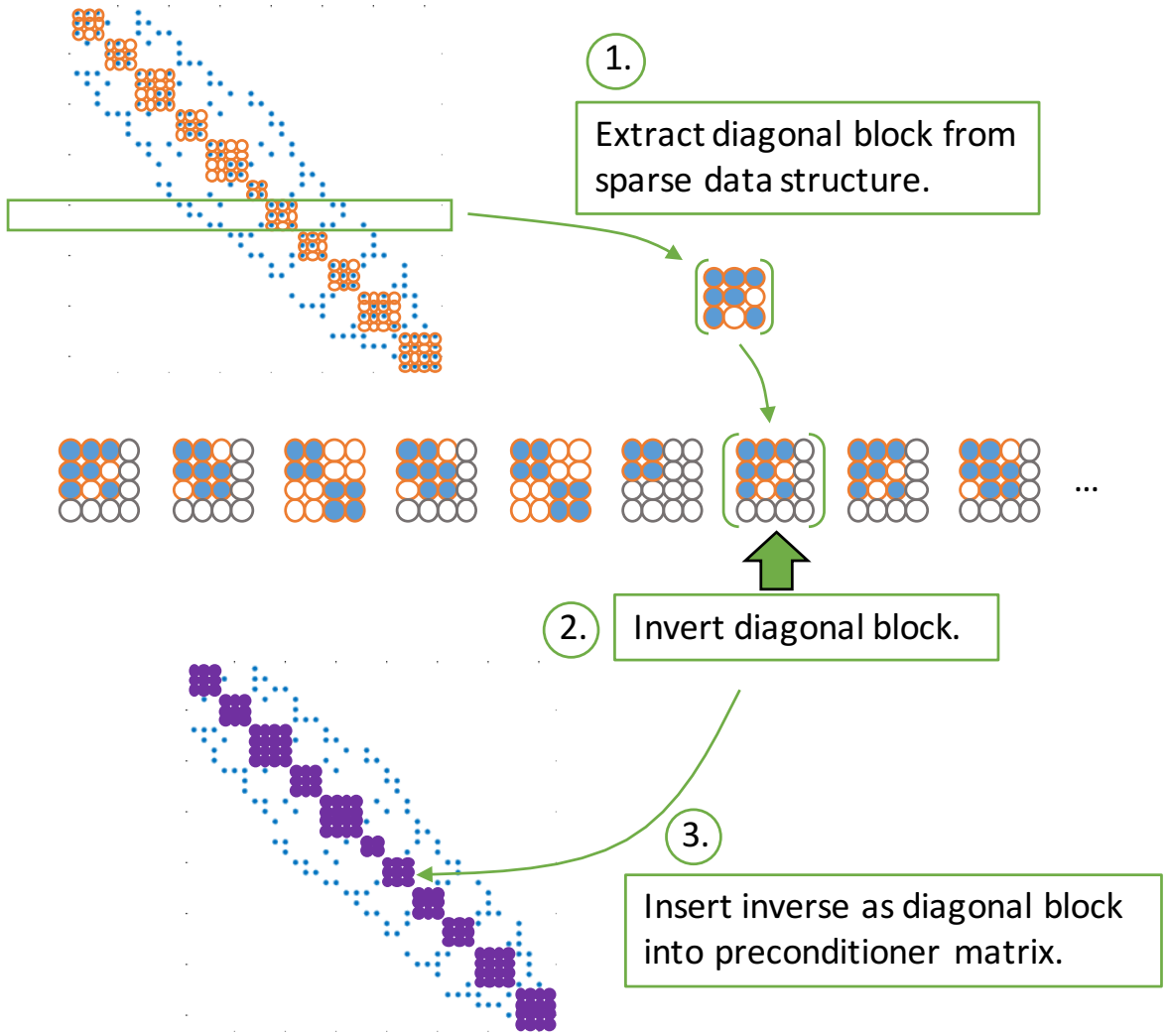
- Cost of **Preconditioner application**:
  m_i^2 FLOPs for block of size m_i :

$$\sum_{blocks} m_i^2$$

- Total memory consumption:

$$\sum_{blocks} m_i^2$$

Inversion-based Block-Jacobi Preconditioning



- Cost of **Inversion:**
  $2m_i^3$ FLOPs for block of size m_i.

$$\sum_{blocks} 2m_i^3$$

- Cost of **Preconditioner application**:
  m_i^2 FLOPs for block of size m_i :

$$\sum_{blocks} m_i^2$$

- Total memory consumption:

$$\sum_{blocks} m_i^2$$

- **Energy balance** for one **preconditioner application** (DP)*:

$$\sum_{blocks} m_i^2 \cdot \underbrace{(100}_{\text{Computation/2}} + \underbrace{4800)}_{\text{data read}}$$

*John Shalf (LBNL)

Mixed Precision Block-Jacobi Preconditioning

Mixed Precision Idea:

- Do **all calculations in working precision**
- **Store** the block-Jacobi matrix in **reduced precision**
  - Benefit from **faster data access**
  - Benefit from **reduced data read cost**

Implications:

- Reduced preconditioner quality
  - Need for **more Krylov solver iterations**
  - Potential loss of regularity (breakdown)

- Cost of **Inversion:**
  $2m_i^3$ FLOPs for block of size m_i.

$$\sum_{blocks} 2m_i^3$$

- Cost of **Preconditioner application**:
  m_i^2 FLOPs for block of size m_i :

$$\sum_{blocks} m_i^2$$

- Total memory consumption:

$$\sum_{blocks} m_i^2$$

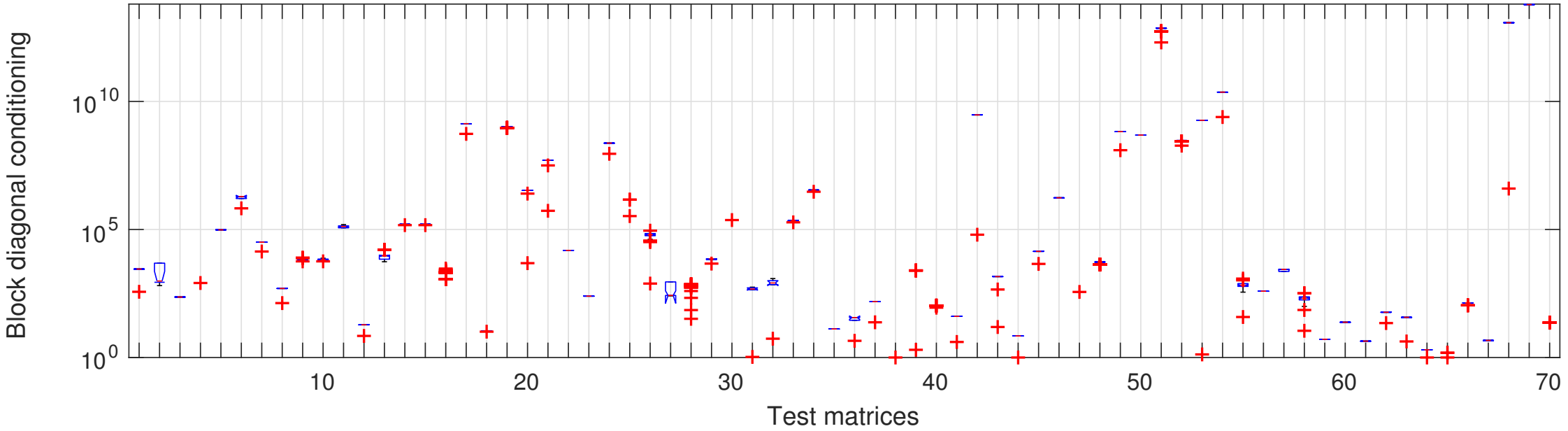
- **Energy balance** for one **preconditioner application** (DP)*:

$$\sum_{blocks} m_i^2 \cdot \underbrace{(100}_{\text{Computation/2}} + \underbrace{4800)}_{\text{data read}}$$

*John Shalf (LBNL)

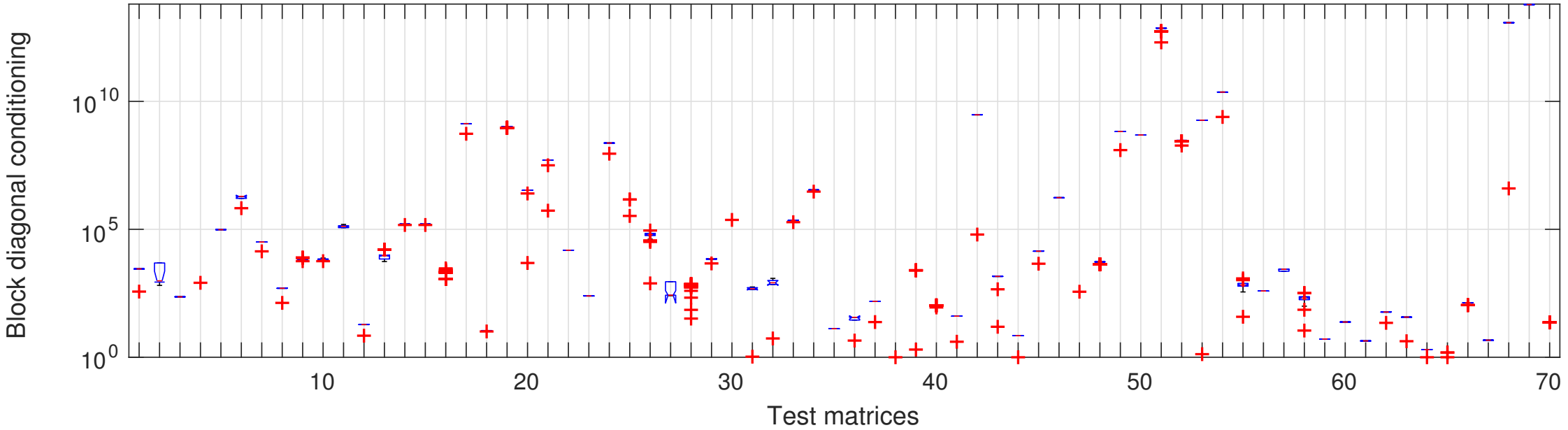
Mixed Precision Block-Jacobi Preconditioning

- 70 matrices from the SuiteSparse Matrix Collection
- Use block-size 24 with Super-Variable agglomeration (24 is upper bound for size of blocks)
- Report conditioning of all arising diagonal blocks

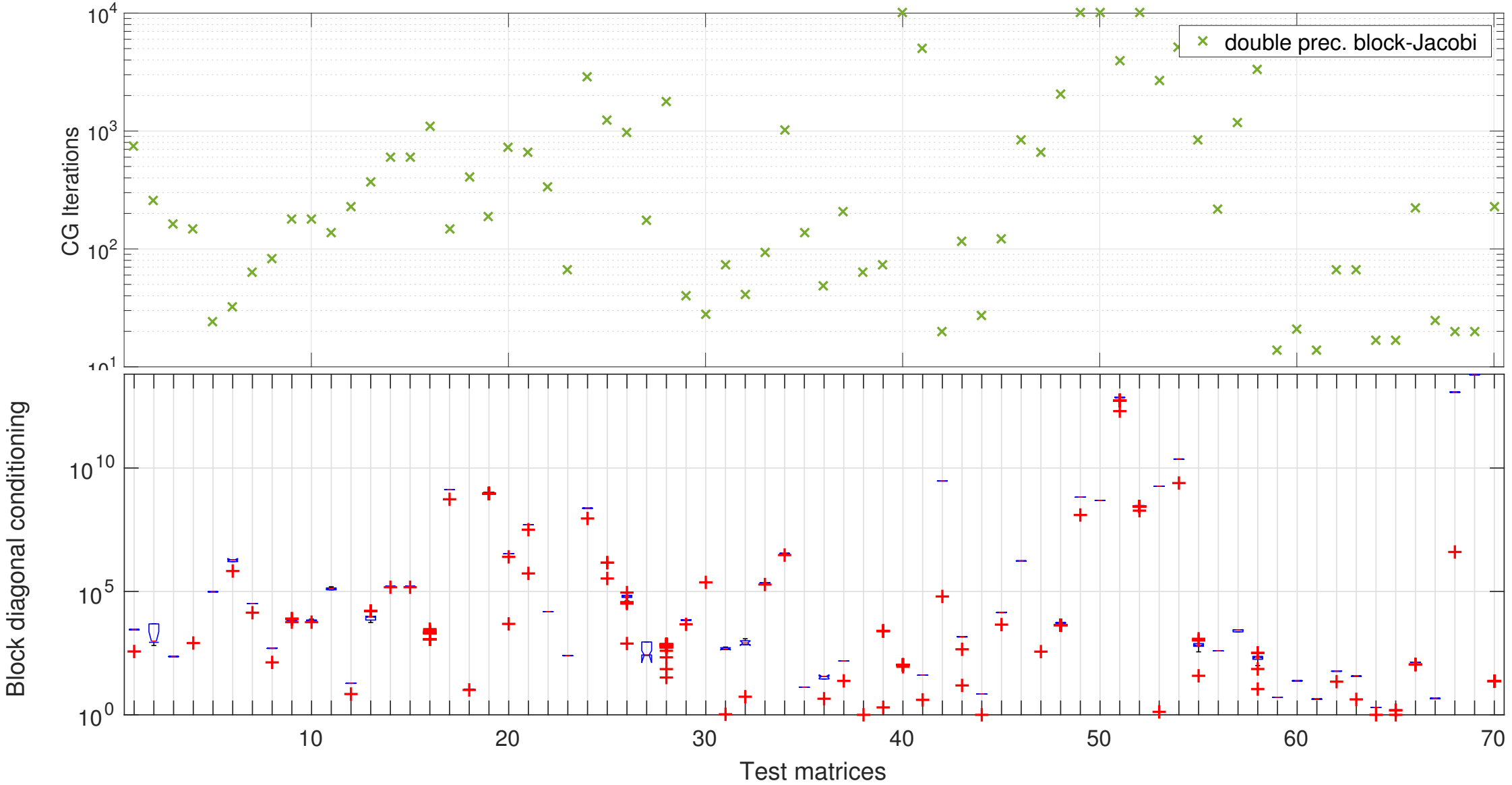


Mixed Precision Block-Jacobi Preconditioning

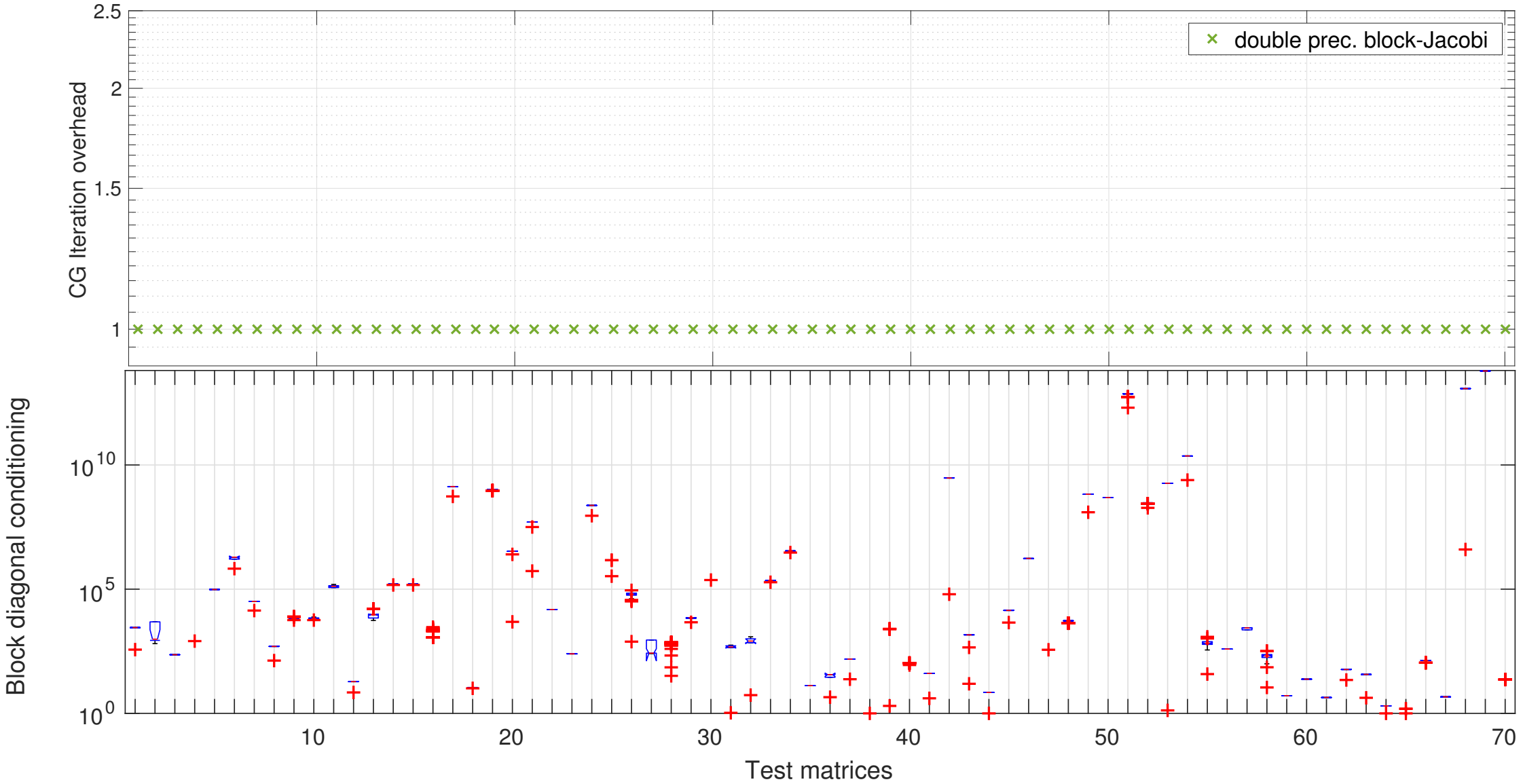
- 70 matrices from the SuiteSparse Matrix Collection
- Use block-size 24 with Super-Variable agglomeration (24 is upper bound for size of blocks)
- Report conditioning of all arising diagonal blocks
- Analyze the impact on a top-level Conjugate Gradient solver (CG)



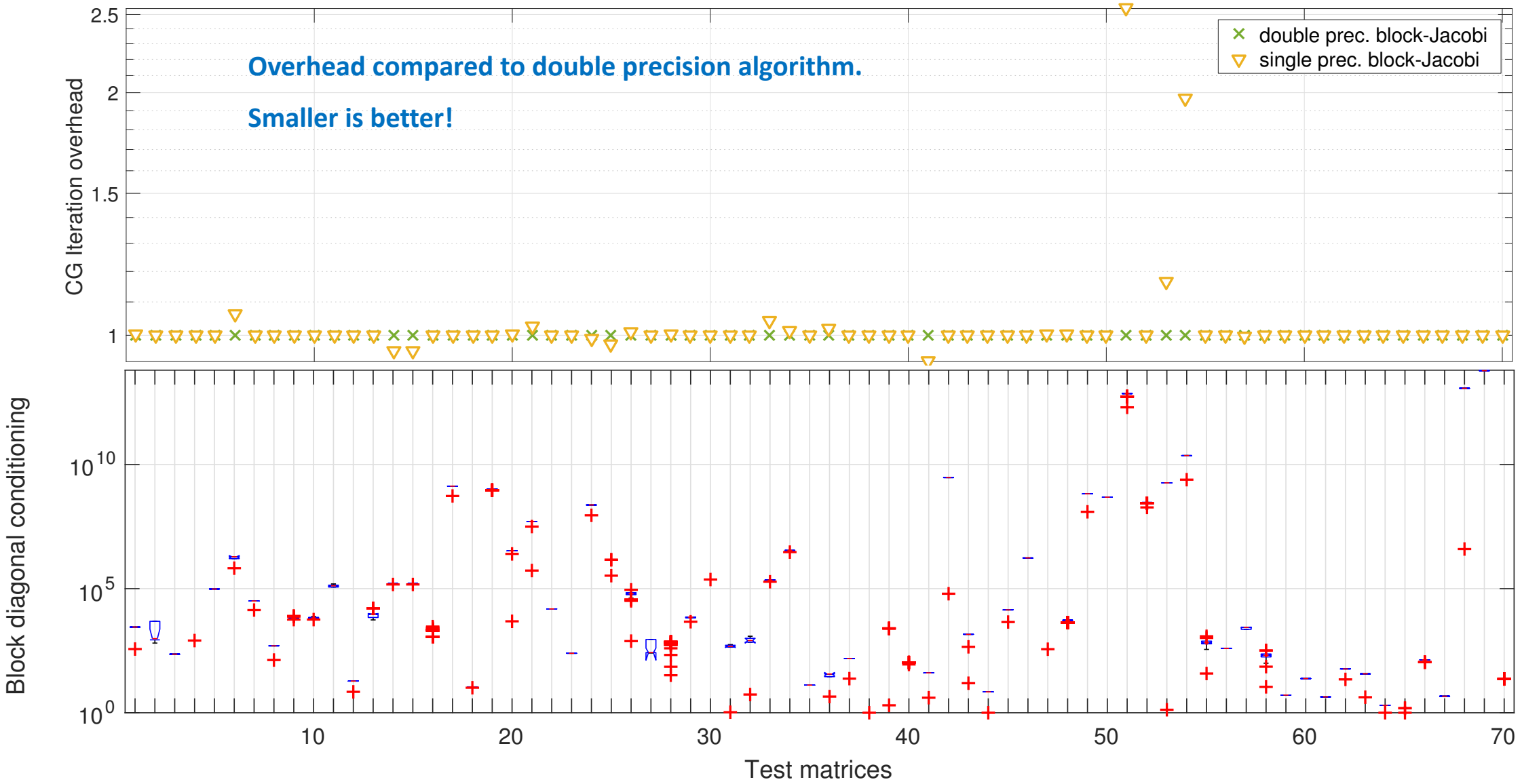
Mixed Precision Block-Jacobi Preconditioning



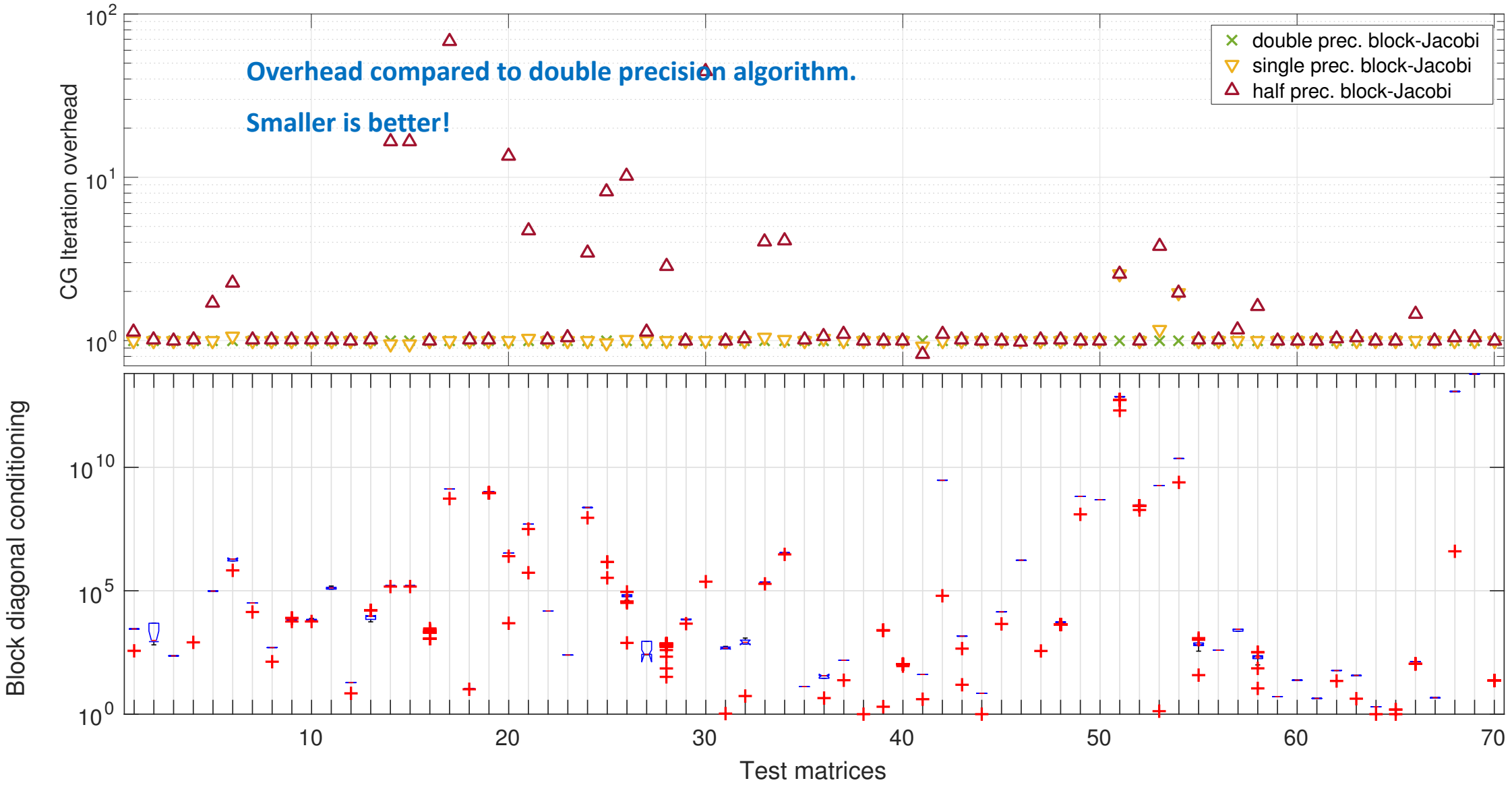
Mixed Precision Block-Jacobi Preconditioning



Mixed Precision Block-Jacobi Preconditioning



Mixed Precision Block-Jacobi Preconditioning

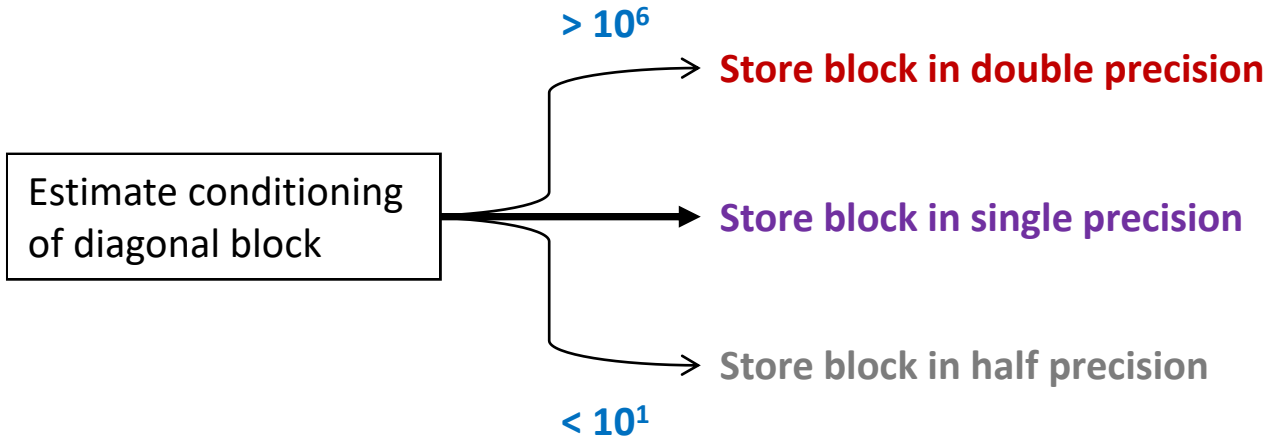


Adaptive Precision Block-Jacobi Preconditioning

Adaptive Precision Idea:

- **All computations use double precision!**

- Store distinct blocks in different formats
- Use **single precision as standard** storage format
- Where **necessary:** switch to **double**
- For well-conditioned blocks use **half precision**

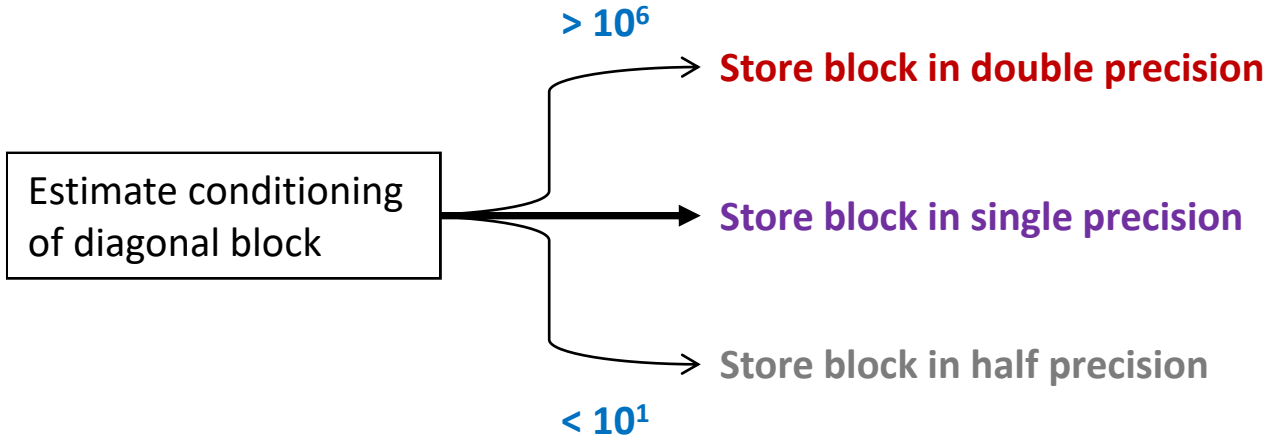


Adaptive Precision Block-Jacobi Preconditioning

Adaptive Precision Idea:

- **All computations use double precision!**

- Store distinct blocks in different formats
- Use **single precision as standard** storage format
- Where **necessary:** switch to **double**
- For well-conditioned blocks use **half precision**



Operation	approximate energy cost
DP floating point multiply-add	100 pJ
DP DRAM read-to-register	4800 pJ
DP word transmit-to-neighbor	7500 pJ
DP word transmit-across-system	9000 pJ

*John Shalf (LBNL)

Energy model:

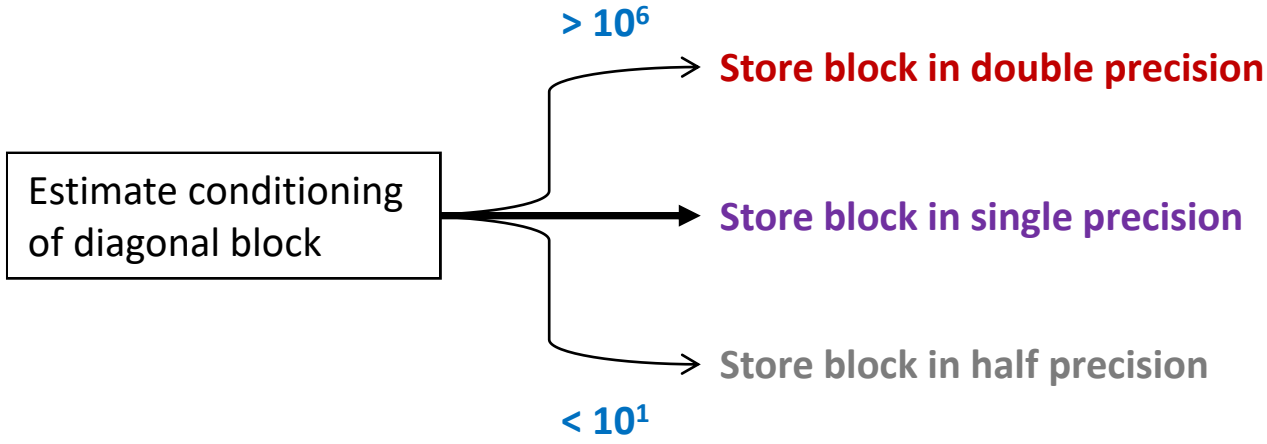
- 4800 pJ for double precision (64 bit)
- 2400 pJ for single precision / integers (32 bit)
- 1200 pJ for half precision (16 bit)

Adaptive Precision Block-Jacobi Preconditioning

Adaptive Precision Idea:

- **All computations use double precision!**

- Store distinct blocks in different formats
- Use **single precision as standard** storage format
- Where **necessary:** switch to **double**
- For well-conditioned blocks use **half precision**



How much data we need to read/write in a Conjugate Gradient (CG) loop:

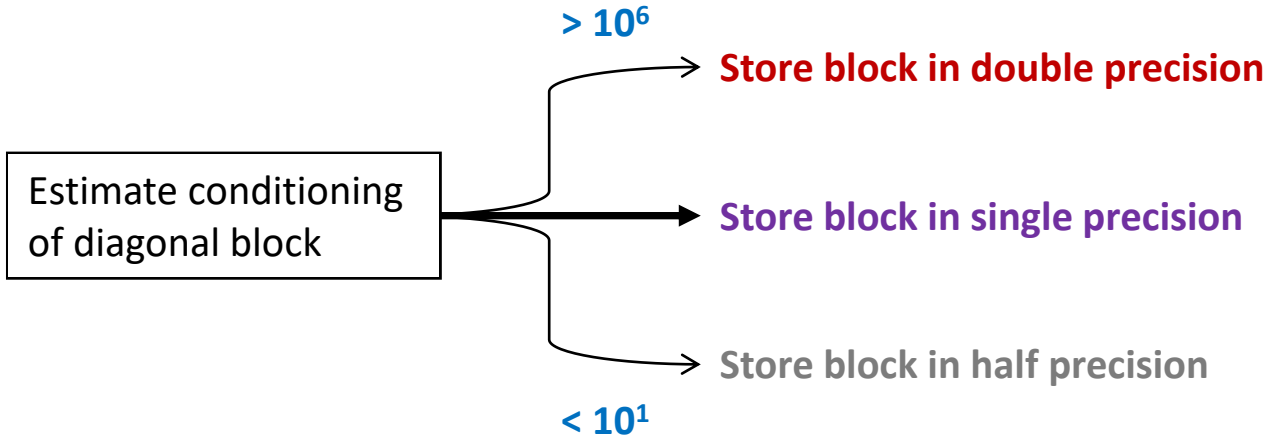
Operation	Memory volume	# per CG loop	Energy est.
CSR-SpMV	nz double + nz int + n int + 2n double	1	(nz + 2n) * 4800 pJ + (nz + n) * 2400 pJ
axpy	3n double	3	9n * 4800 pJ
dot	2n double	2	4n * 4800 pJ
preconditioner	[used format]	1	* ?

Adaptive Precision Block-Jacobi Preconditioning

Adaptive Precision Idea:

- **All computations use double precision!**

- Store distinct blocks in different formats
- Use **single precision as standard** storage format
- Where **necessary:** switch to **double**
- For well-conditioned blocks use **half precision**



Energy model for block-Jacobi-CG

- We ignore computational cost, only memory access
- No data (matrix / vector) is cached, only DRAM reads
- CG outer solver (in DP)
- Adaptive precision for the block-Jacobi preconditioner

Block-Jacobi CG

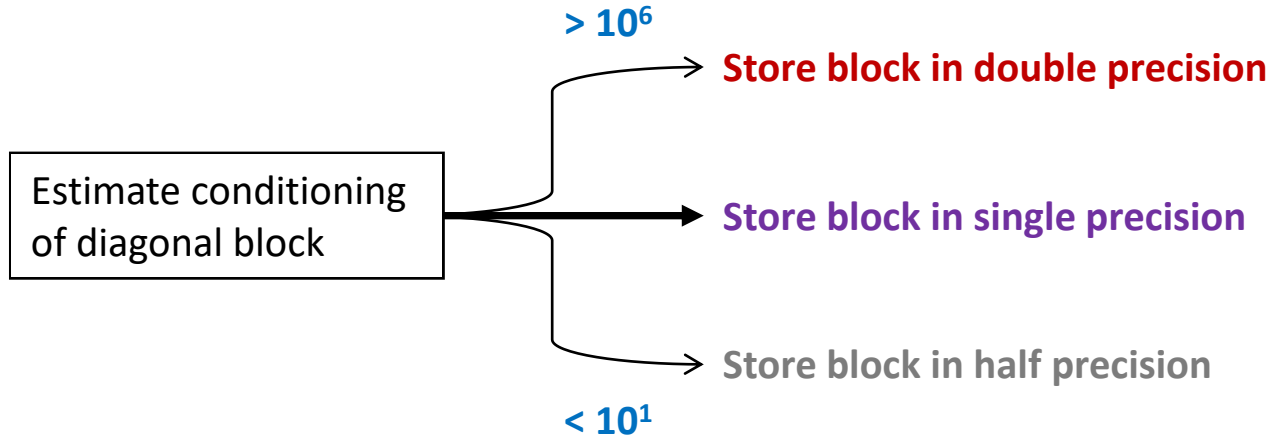
- Ax=b with x:=0 and b:=A*1
- Relative residual stopping crit. 1e-9

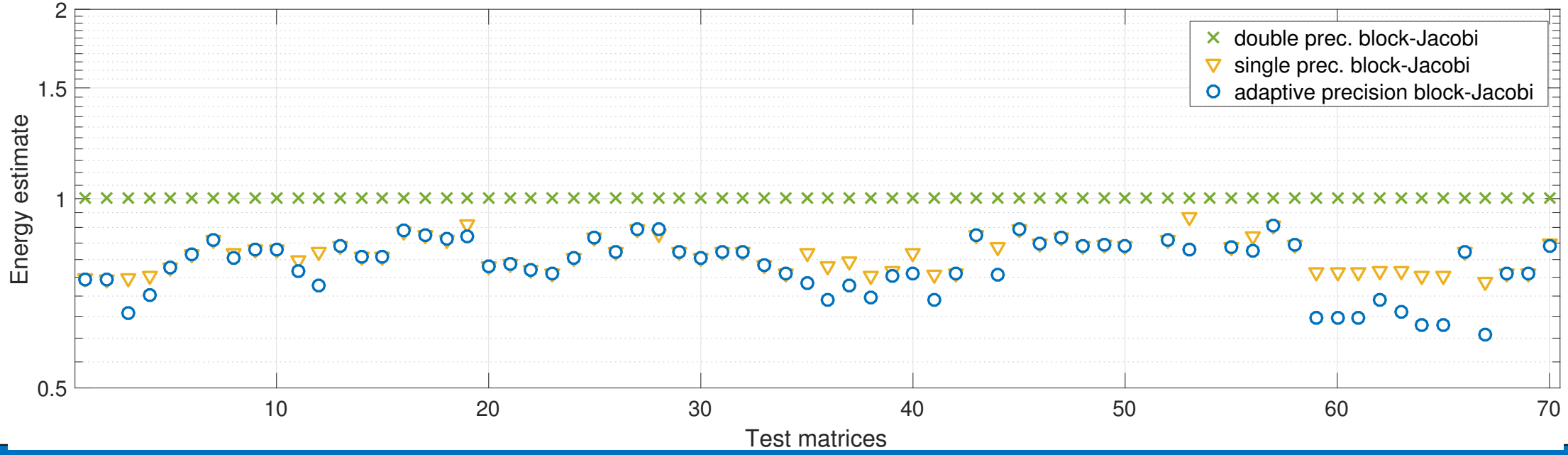
Adaptive Precision Block-Jacobi Preconditioning

Adaptive Precision Idea:

- **All computations use double precision!**

- Store distinct blocks in different formats
- Use **single precision as standard** storage format
- Where **necessary:** switch to **double**
- For well-conditioned blocks use **half precision**





Take-Away Message

- **Data access extremely important in energy balance!**
- **FLOPs** are **less important.**
- Tolerate additional FLOPs for reduced data access.
  - Storing information in lower than working precision.
  - Storing information in a compressed fashion (C. Penke).
- **Preconditioners** are typically only a rough **approximation.**
  - **Reducing precision** can be **acceptable** without harming convergence.
  - Trade-off between preconditioner accuracy and outer solver iterations.
- **Adaptive Precision methods focusing on reduced storage can improve energy balance.**

This research is based on a cooperation between Hartwig Anzt (KIT, UTK), Jack Dongarra (University of Tennessee, ORNL), Goran Flegar and Enrique S. Quintana-Ortí (University Jaume I), and Nick Higham (The University of Manchester).