Science & Technology Facilities Council

Pivoting for Matrix Factorizations on Manycore Architectures

Jonathan Hogg

STFC Rutherford Appleton Laboratory

4 September 2014
IMA Numerical Linear Algebra and Optimization
Birmingham, UK

Manycore

Chip	Cores	GB/ sec	TFLOP/ sec	GFLOPS/ Watt
NVIDIA K40	15×64	288	1.43	6.1
NVIDIA Titan Z	$2 \times 15 \times 64$	672	2.66	7.1
AMD R9 295X2	$2 \times 44 \times 8^*$	640	1.43	2.9
Intel Xeon Phi	60×8	320	1.00	4.5
Intel Desktop E5-2687W	16×4	50	0.40	1.3

* double precision cores. single precision is $8\times$.

Science & Technology Facilities Council

Sources of parallelism

Need to get chips "firing on all cylinders":

- Extra-core shared memory parallelism between physical cores
- Intra-core shared memory parallelism between hyperthreads on a physical core
- Vectorization within a physical core (currently up to length 32)
- Instruction-level Parallelism (ILP) to keep functional units busy

Plus multiprocessor parallelism (multiple sockets or MPI)

Example

Recent Intel server

- 2 sockets each with E5-2687W @ 3.10GHz (Sandy Bridge-EP)
- Each has 8 physical cores
- Each core has two decoder streams (hyperthreads)
- Each core has a 256-bit vector unit (4 doubles/8 singles)
- Can issue one vector add and one vector mul per cycle
- 3-5 cycle instruction latency

Example

Recent Intel server

- 2 sockets each with E5-2687W @ 3.10GHz (Sandy Bridge-EP)
- Each has 8 physical cores
- Each core has two decoder streams (hyperthreads)
- Each core has a 256-bit vector unit (4 doubles/8 singles)
- Can issue one vector add and one vector mul per cycle
- 3-5 cycle instruction latency

Naive code (single core, dbl prec)	0.52%	2 GFlop/s
+Full ILP	1.56%	6 GFlop/s
+Vectorization	6.25%	25 GFlop/s
+Shmem parallel	100.00%	400 GFlop/s

Science & Technology Facilities Council

Example

Recent Intel server

- 2 sockets each with E5-2687W @ 3.10GHz (Sandy Bridge-EP)
- Each has 8 physical cores
- Each core has two decoder streams (hyperthreads)
- Each core has a 256-bit vector unit (4 doubles/8 singles)
- Can issue one vector add and one vector mul per cycle
- 3-5 cycle instruction latency

Naive code (single core, dbl prec)	0.52%	2 GFlop/s
+Full ILP	1.56%	6 GFlop/s
+Vectorization	6.25%	25 GFlop/s
+Shmem parallel	100.00%	400 GFlop/s

Intel Pentium 4 @ 3GHz (2004) = 12 GFlop/s (128 bit vector)
Intel Pentium 75 (1994) = 0.075 GFlop/s

Science & Technology Facilities Council

Our problem

Solve $Ax = b$ for sparse A.
Fast.

- Consider $A = LDL^T$ for symmetric indefinite systems
- ...but most things can be generalised to LU.
- Numerical stability is important
  - Normally control growth through pivoting
  - ...basically avoiding division by small numbers
- Traditional techniques fail when you need to keep >2000 threads busy at once for "small" problems

Massive parallelism

So how do I keep 2000 units busy at once?

- Traditional tree-based approach doesn't scale
- ...and might be used for inter-node parallelism anyway
- Threaded BLAS insufficient, especially for small matrices

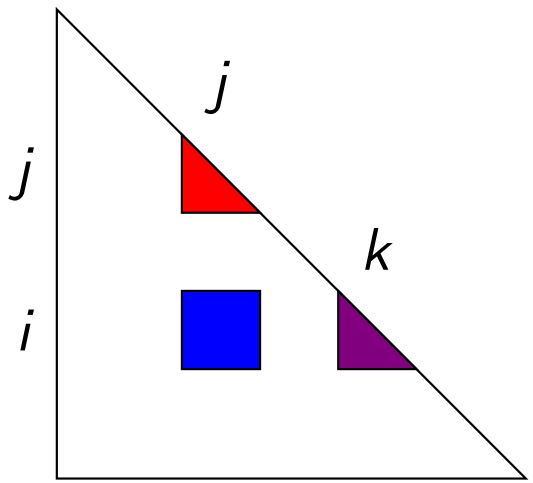
Maximize throughput

- Find fine-grained parallelism in dense kernels
- Avoid bottlenecks
- Latency often critical for small problems
  - Inherent in traditional pivoting
  - (can't progress until decision made)

LDL^T factorization refresher

Work by blocks:

- Factorize dense blocks on diagonal using dense algorithm $A_{jj} = L_{jj} D_{jj} L_{jj}^T$
- "Divide" remainder of column by diagonal block $L_{ij} = A_{ij} L_{jj}^{-T}$
- Update matrix to right as $A_{ik} = A_{ik} - L_{ij} D_{jj} L_{kj}^T$



Science & Technology Facilities Council

Copy approaches from dense case?

Take task based approach (e.g. PLASMA/MAGMA/StarPU)

- Divide large matrix coarsely into blocks
  - Big enough to get good BLAS performance
- Data dependencies via Directed Acyclic Graph (DAG)
- Handle pivoting differently:
  - Externally through RBM transforms [Parker 1995]
  - Tournament Pivoting [Grigori et al. 2008]

Copy approaches from dense case?

Take task based approach (e.g. PLASMA/MAGMA/StarPU)

- Divide large matrix coarsely into blocks
  - Big enough to get good BLAS performance
- Data dependencies via Directed Acyclic Graph (DAG)
- Handle pivoting differently:
  - Externally through RBM transforms [Parker 1995]
  - Tournament Pivoting [Grigori et al. 2008]

HSL_MA86/7 (sparse symmetric) implemented this **Problems:**

- Matrix already blocked on a fine level
  - Often not large enough to hit full vectorization in BLAS
- Finer blocks $\Rightarrow$ much bigger DAGs
- Pivoting approaches don't work
  - RBM have to choose between density and reliability (10$\times$ denser for 90% reliable [Baboulin et al. 2014])
  - Limiting pivots near diagonal breaks tournament pivoting

Science & Technology Facilities Council

Solutions

Fine blocks

- Live with it: no worse than previous approaches
- More aggressive supernode amalgamation
- Specialised dense/sparse in-cache kernels (work in progress)

Solutions

Fine blocks

- Live with it: no worse than previous approaches
- More aggressive supernode amalgamation
- Specialised dense/sparse in-cache kernels (work in progress)

Bigger DAGs

- Use implicit representation [Hogg et al. 2010]
- Nested DAGs [Kim and Eijkhout 2014]
- Delegate the problem to a scheduling library

Solutions

Fine blocks

- Live with it: no worse than previous approaches
- More aggressive supernode amalgamation
- Specialised dense/sparse in-cache kernels (work in progress)

Bigger DAGs

- Use implicit representation [Hogg et al. 2010]
- Nested DAGs [Kim and Eijkhout 2014]
- Delegate the problem to a scheduling library

Pivoting

- Subject of the remainder of this talk

Traditional Threshold Partial Pivoting: Test

For backwards stability:

- Sufficient to bound entries of L
- Threshold test such that $L_{ij} = A_{ij} L_{jj}^{-T}$ yields $l_{ij} \leq u^{-1}$
- Need to consider whole column

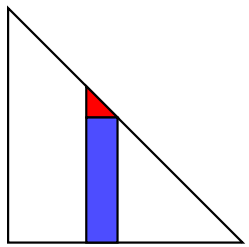
Traditional Threshold Partial Pivoting: Test

For backwards stability:

- Sufficient to bound entries of L
- Threshold test such that $L_{ij} = A_{ij} L_{jj}^{-T}$ yields $l_{ij} \leq u^{-1}$
- Need to consider whole column

1×1 pivot test

$$|a_{jj}| \geq u \max_{i>j} |a_{ij}|$$



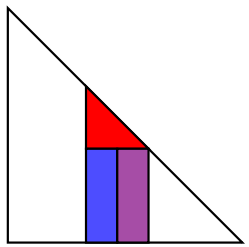
Traditional Threshold Partial Pivoting: Test

For backwards stability:

- Sufficient to bound entries of L
- Threshold test such that $L_{ij} = A_{ij} L_{jj}^{-T}$ yields $l_{ij} \leq u^{-1}$
- Need to consider whole column

1×1 pivot test

$$|a_{jj}| \geq u \max_{i>j} |a_{ij}|$$

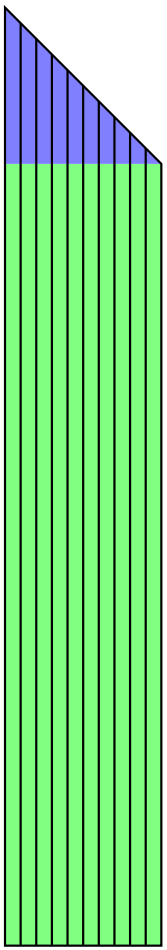


2×2 pivot test

$$\left| \begin{pmatrix} a_{jj} & a_{j(j+1)} \\ a_{j(j+1)} & a_{(j+1)(j+1)} \end{pmatrix}^{-1} \right| \begin{pmatrix} \max_{i>j+1} |a_{ij}| \\ \max_{i>j+1} |a_{i(j+1)}| \end{pmatrix} \leq \begin{pmatrix} u^{-1} \\ u^{-1} \end{pmatrix}$$

Science & Technology Facilities Council

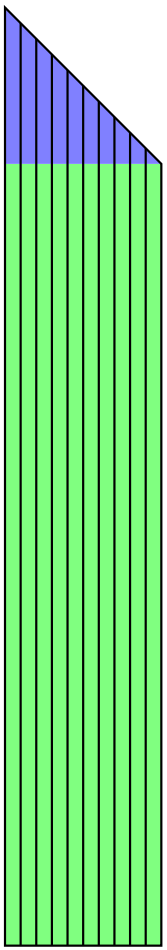
Traditional Threshold Partial Pivoting: Implementation



Traditional algorithm

- Work column by column
- Bring column up-to-date
- Find maximum element α in column of A_{21}
- Pivot test: accept/reject pivot

Traditional Threshold Partial Pivoting: Implementation



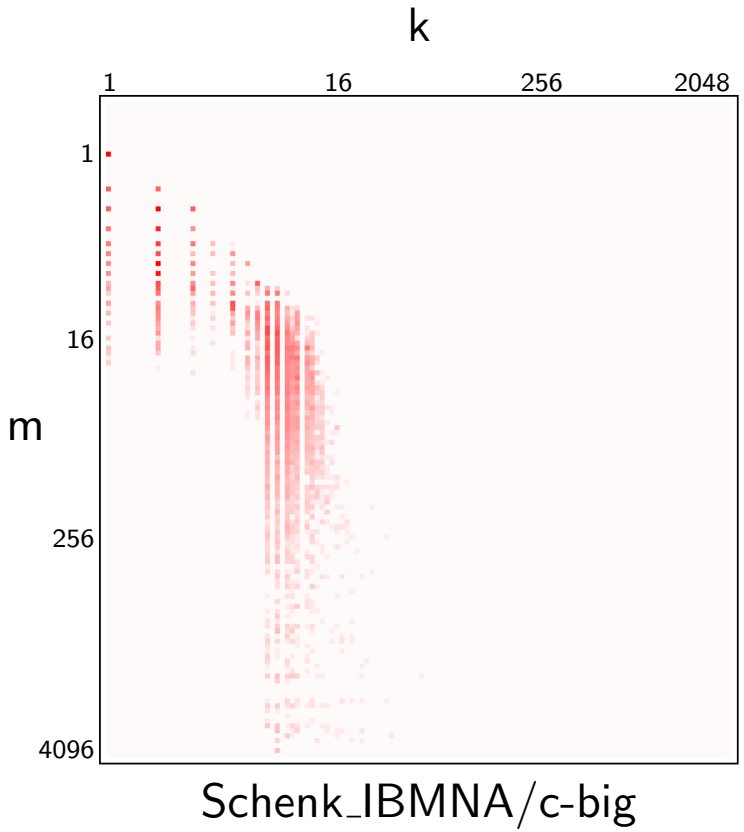
Traditional algorithm

- Work column by column
- Bring column up-to-date
- Find maximum element α in column of A_{21}
- Pivot test: accept/reject pivot

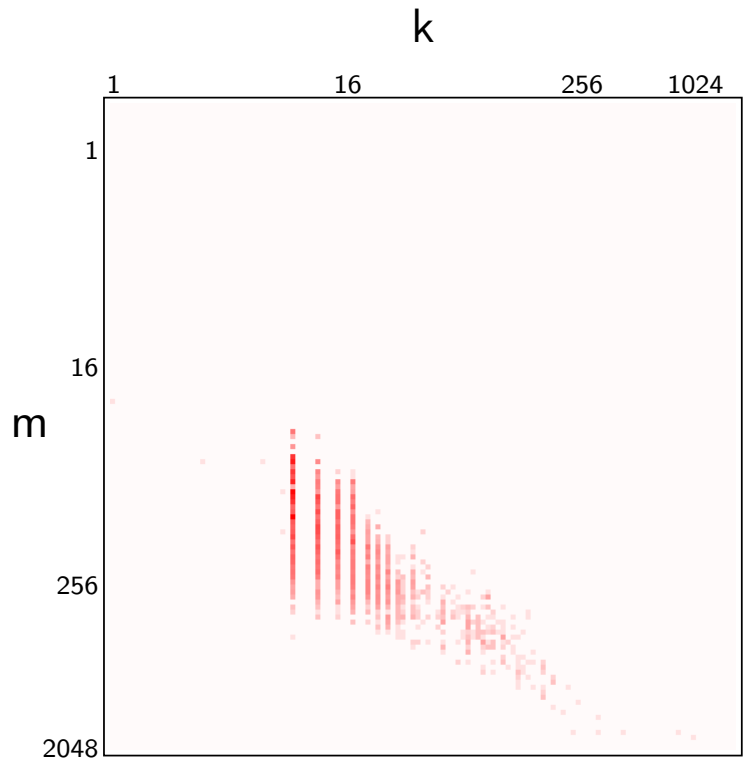
Problems

- Very stop-start (one column at a time)
- All-to-all communication for every column

Size distributions



Schenk_IBMNA/c-big



GHS_psdef/bmwcra_1

- Wide range of sizes
- Often $m \gg k$

Science & Technology Facilities Council

Do we actually need pivoting?

Observation

- Many problems don't actually need pivoting
- Especially if scaling and IR are used

Exploitation

- Just use a Cholesky-like framework to compute LDL^T
- Optionally restrict pivoting to blocks on the diagonal [Schenk and Gärtner 2004]
- Optionally use static pivoting to prevent breakdown [Li and Demmel 1998]

Above + matching-based ordering + IR/FGMRES
Solves >95% problems to machine precision.
Approach used in PARDISO [Schenk et al.]
What about the rest?

Some other approaches

Matching-based ordering [Duff et al. 2005]

- Find large entries in A and force to subdiagonal
- Sometimes cure worse than disease: reduces sparsity

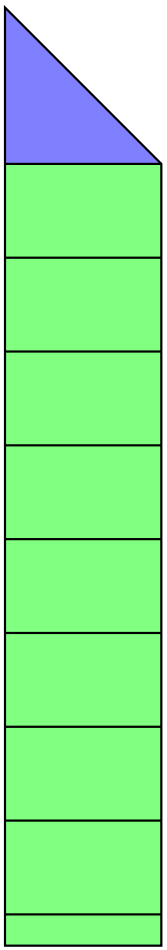
Parallel pivoting [Darron et al. 1960]

- Pivots compared pairwise only (but many eliminated at once)
- Highly unstable

Pairwise pivoting [Sorenson 1985]

- Pivots compared pairwise only (one at a time)
- More stable, but nonlinear growth as matrix size increases

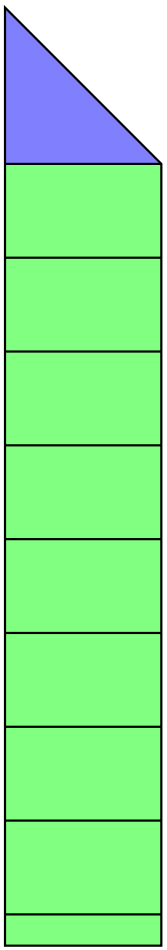
Best of both worlds?



Outline:

Assume no pivoting needed, backtrack if it was

Best of both worlds?



Outline:

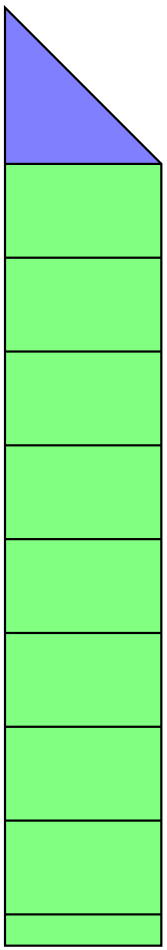
Assume no pivoting needed, backtrack if it was

Try-it-and-see pivoting (a posteriori pivoting)

- Work by blocks of L_{21}
- Every block factorizes copy of A_{11}
- Every block checks $\max |l_{21}| < u^{-1}$
- All-to-all communication when all blocks are done
- Discard columns that have failed on *any* block

Science & Technology Facilities Council

Best of both worlds?



Outline:

Assume no pivoting needed, backtrack if it was

Try-it-and-see pivoting (a posteriori pivoting)

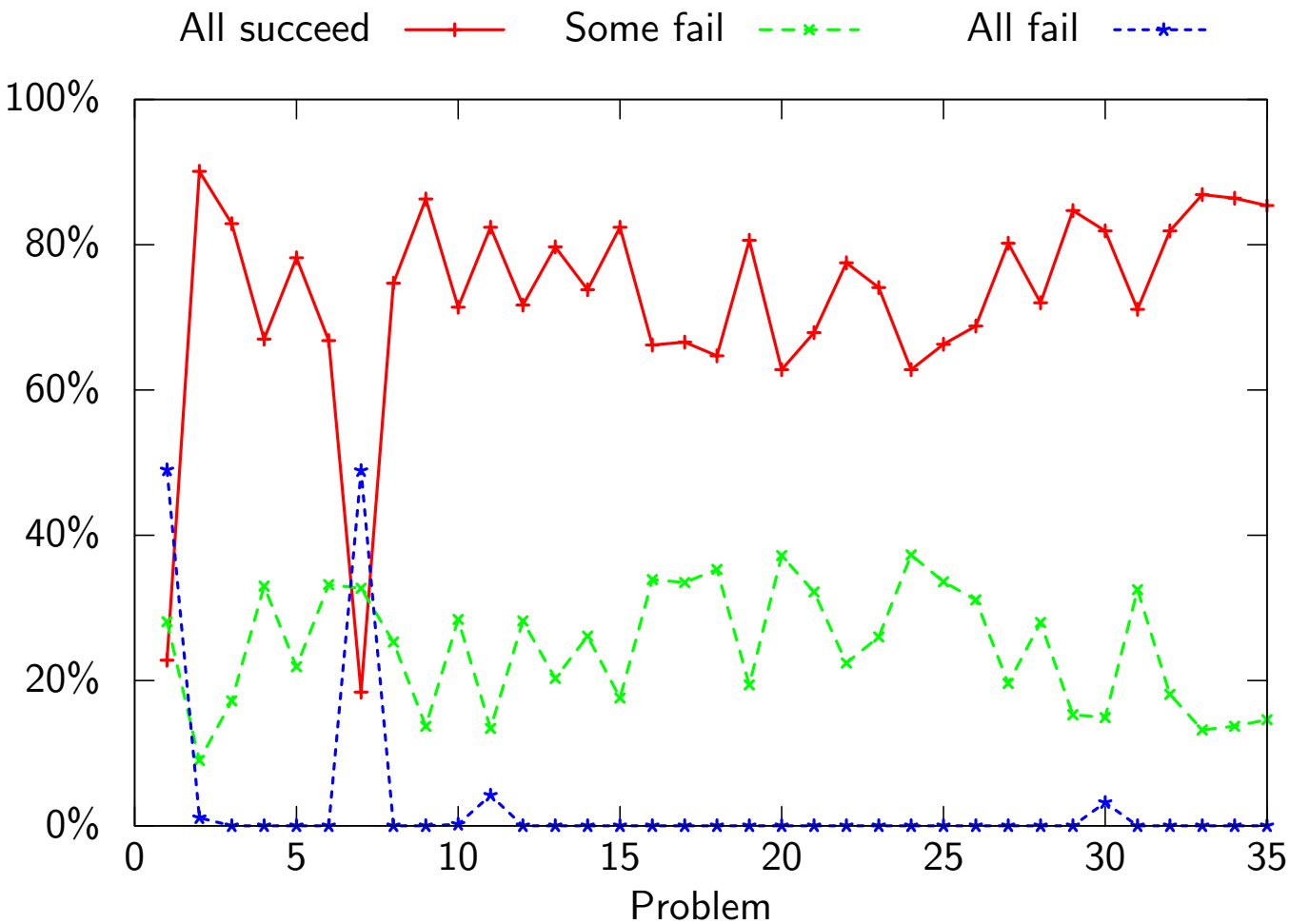
- Work by blocks of L_{21}
- Every block factorizes copy of A_{11}
- Every block checks $\max |l_{21}| < u^{-1}$
- All-to-all communication when all blocks are done
- Discard columns that have failed on *any* block

Implementation Issues

- Inefficient if lots of rejected pivots
- Still quite stop-start

Science & Technology Facilities Council

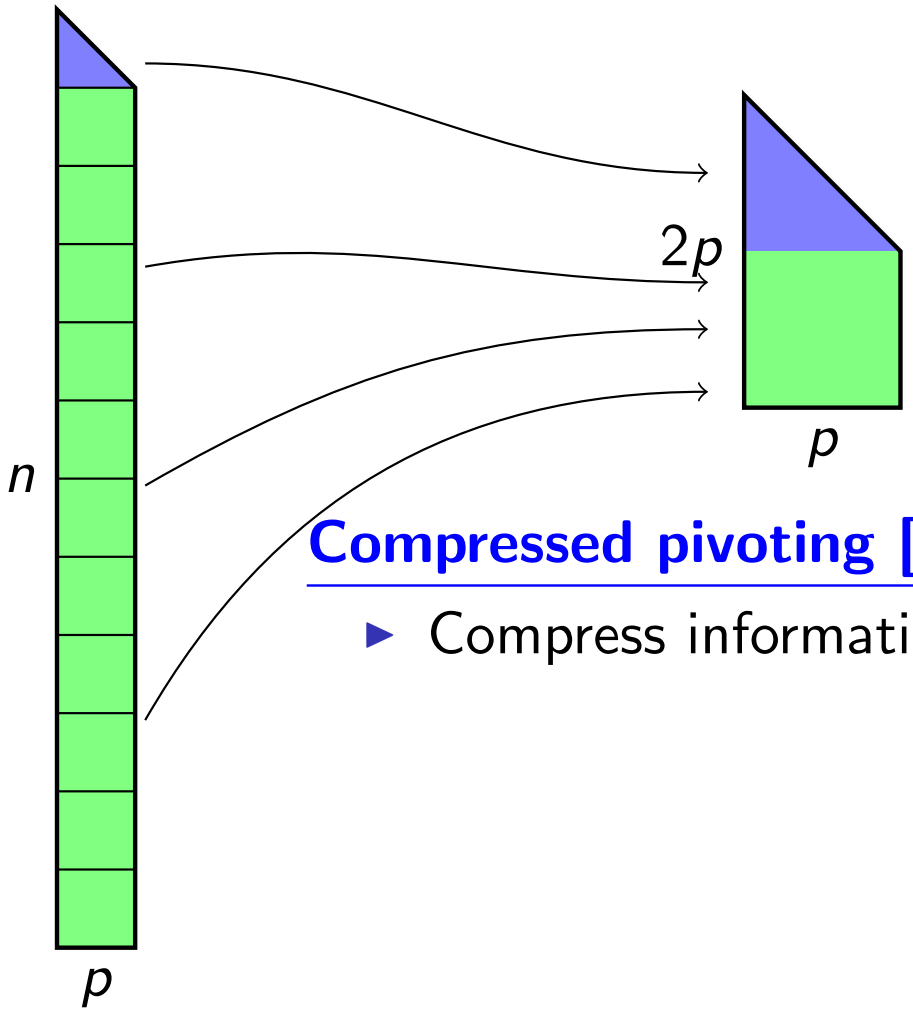
A Posteriori Pivoting Results



- Only problems 1 and 6 need pivoting
- Almost always make *some* progress

Science & Technology Facilities Council

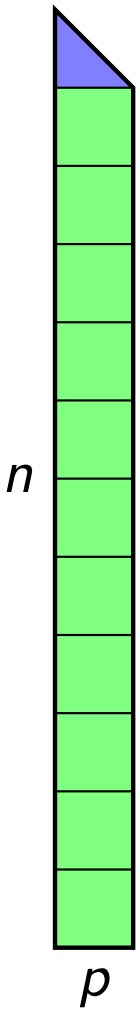
Fallback: Compressed pivoting

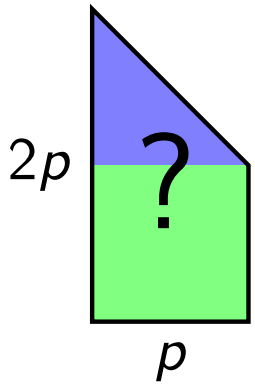


Compressed pivoting [Hogg and Scott 2013]

- Compress information into small matrix

Fallback: Compressed pivoting

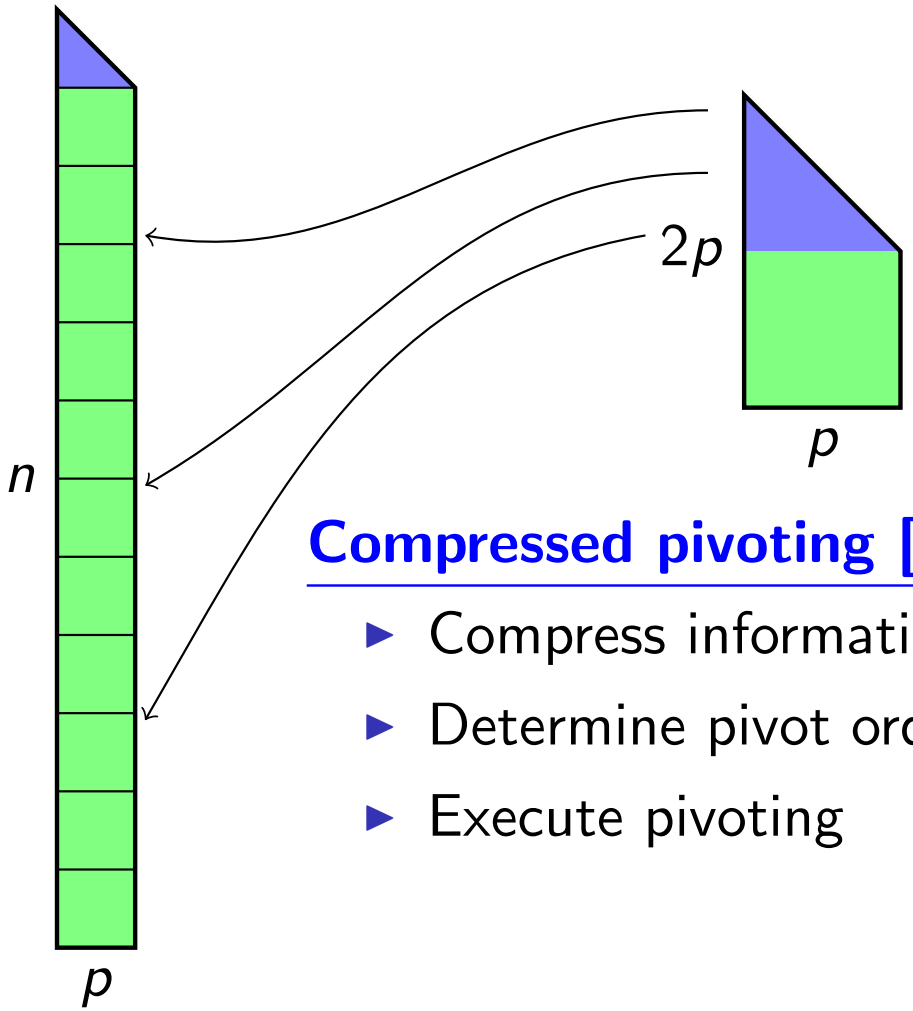




Compressed pivoting [Hogg and Scott 2013]

- Compress information into small matrix
- Determine pivot order

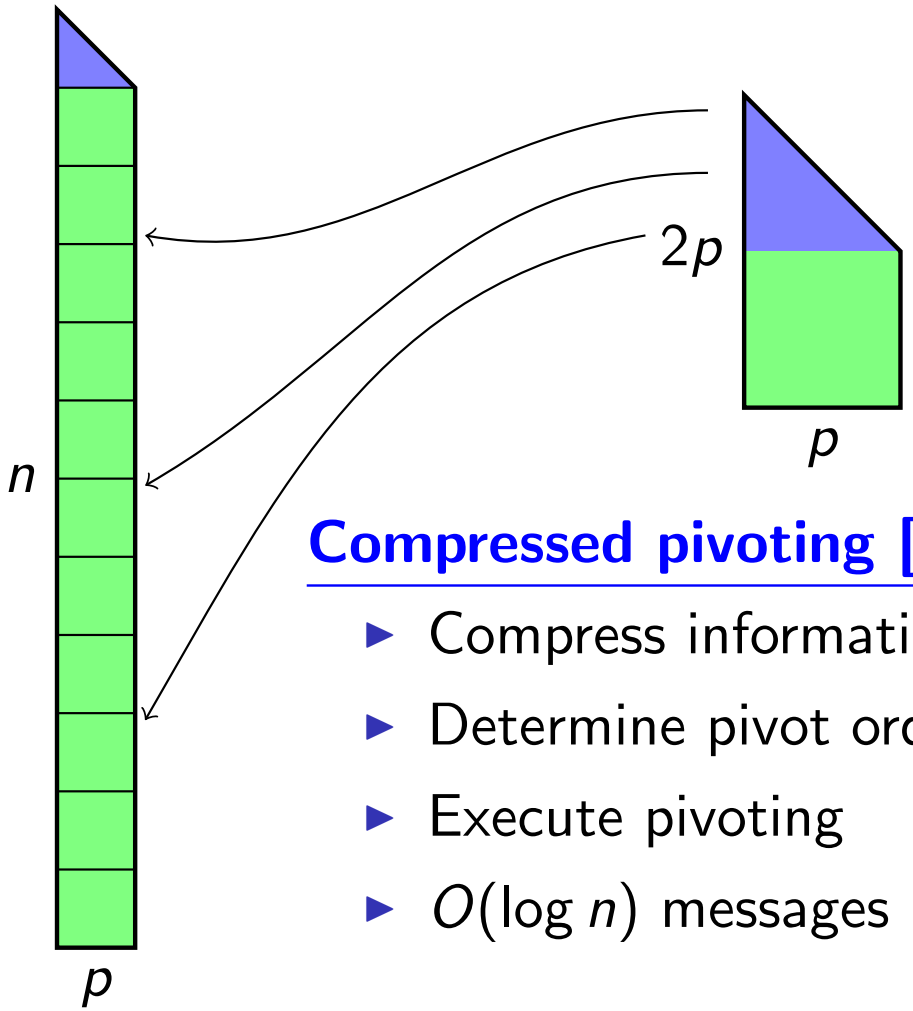
Fallback: Compressed pivoting



Compressed pivoting [Hogg and Scott 2013]

- Compress information into small matrix
- Determine pivot order
- Execute pivoting

Fallback: Compressed pivoting

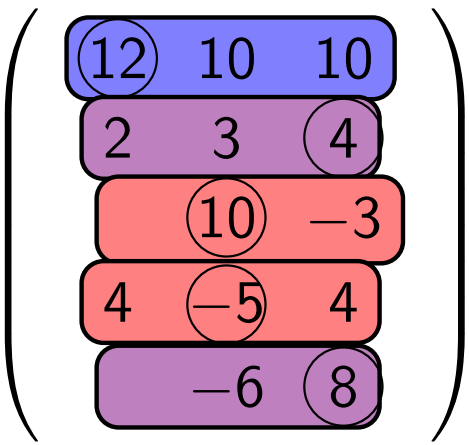


Compressed pivoting [Hogg and Scott 2013]

- Compress information into small matrix
- Determine pivot order
- Execute pivoting
- $O(\log n)$ messages rather than $O(p \log n)$

Strict Compressed Pivoting

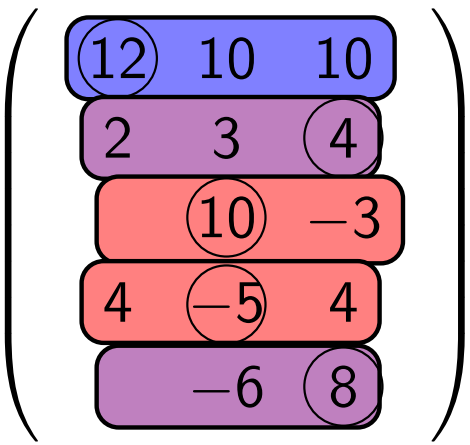
1. Partition rows into sets by column of maximum $|a_{ij}|$



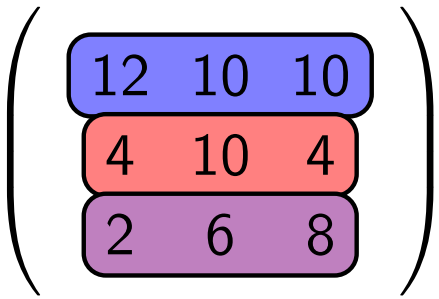
Partitioned rows

Strict Compressed Pivoting

1. Partition rows into sets by column of maximum $|a_{ij}|$
2. Represent each set by single row: take maximum $|a_{ij}|$



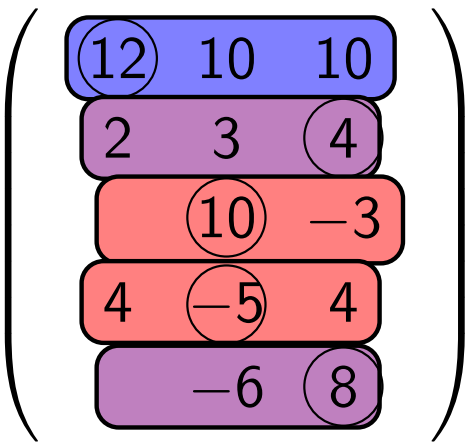
Partitioned rows



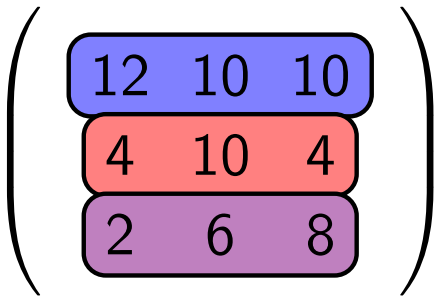
Compressed matrix

Strict Compressed Pivoting

1. Partition rows into sets by column of maximum $|a_{ij}|$
2. Represent each set by single row: take maximum $|a_{ij}|$
3. Update using a "worst-case" formula



Partitioned rows



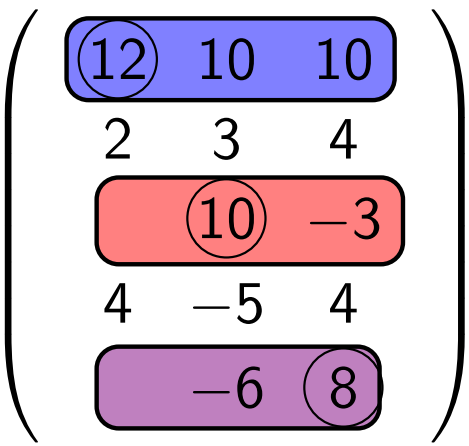
Compressed matrix

- Provably backwards stable
- Sometimes too pessimistic

Science & Technology Facilities Council

Relaxed Compressed Pivoting

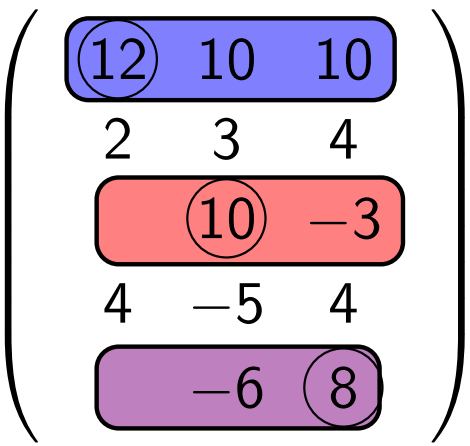
1. For each column, pick a "representative" row: largest $|a_{ij}|$
2. Apply standard threshold partial pivoting.



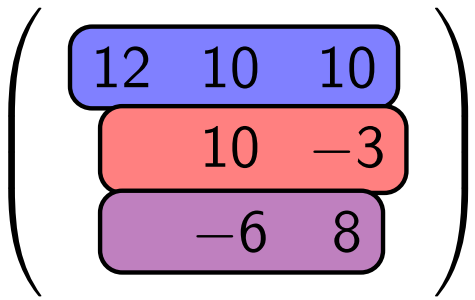
Partitioned rows

Relaxed Compressed Pivoting

1. For each column, pick a "representative" row: largest $|a_{ij}|$
2. Apply standard threshold partial pivoting.



Partitioned rows

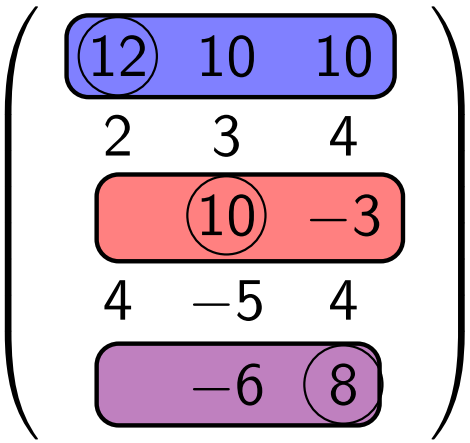


Compressed matrix

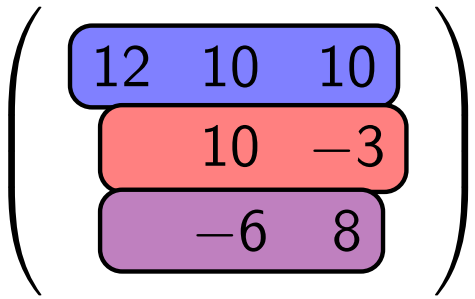
Science & Technology Facilities Council

Relaxed Compressed Pivoting

1. For each column, pick a "representative" row: largest $|a_{ij}|$
2. Apply standard threshold partial pivoting.



Partitioned rows

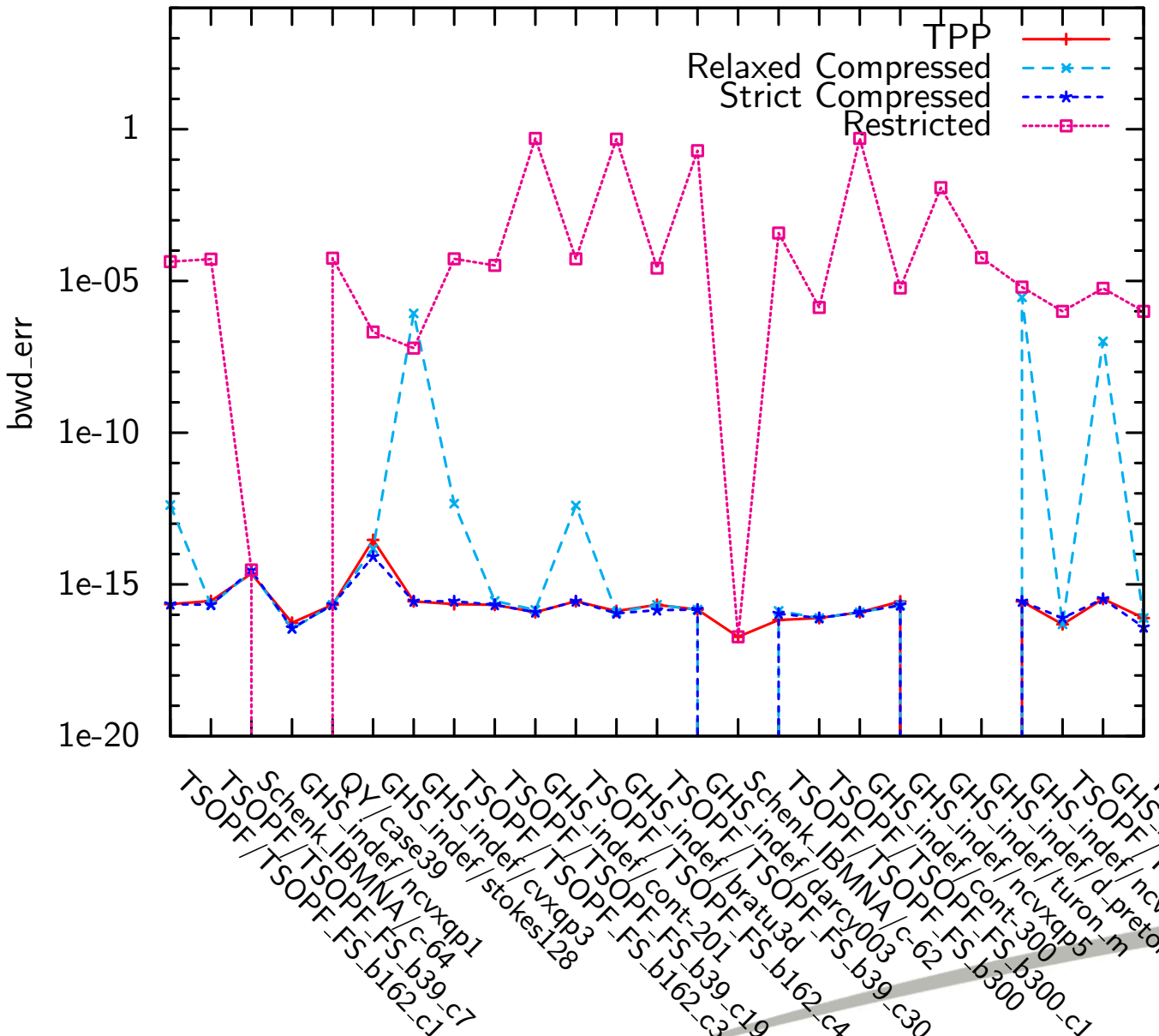


Compressed matrix

- Not backwards stable!
- Stable in practice (see results)

Science & Technology Facilities Council

Compressed Pivoting: Numerical Stability

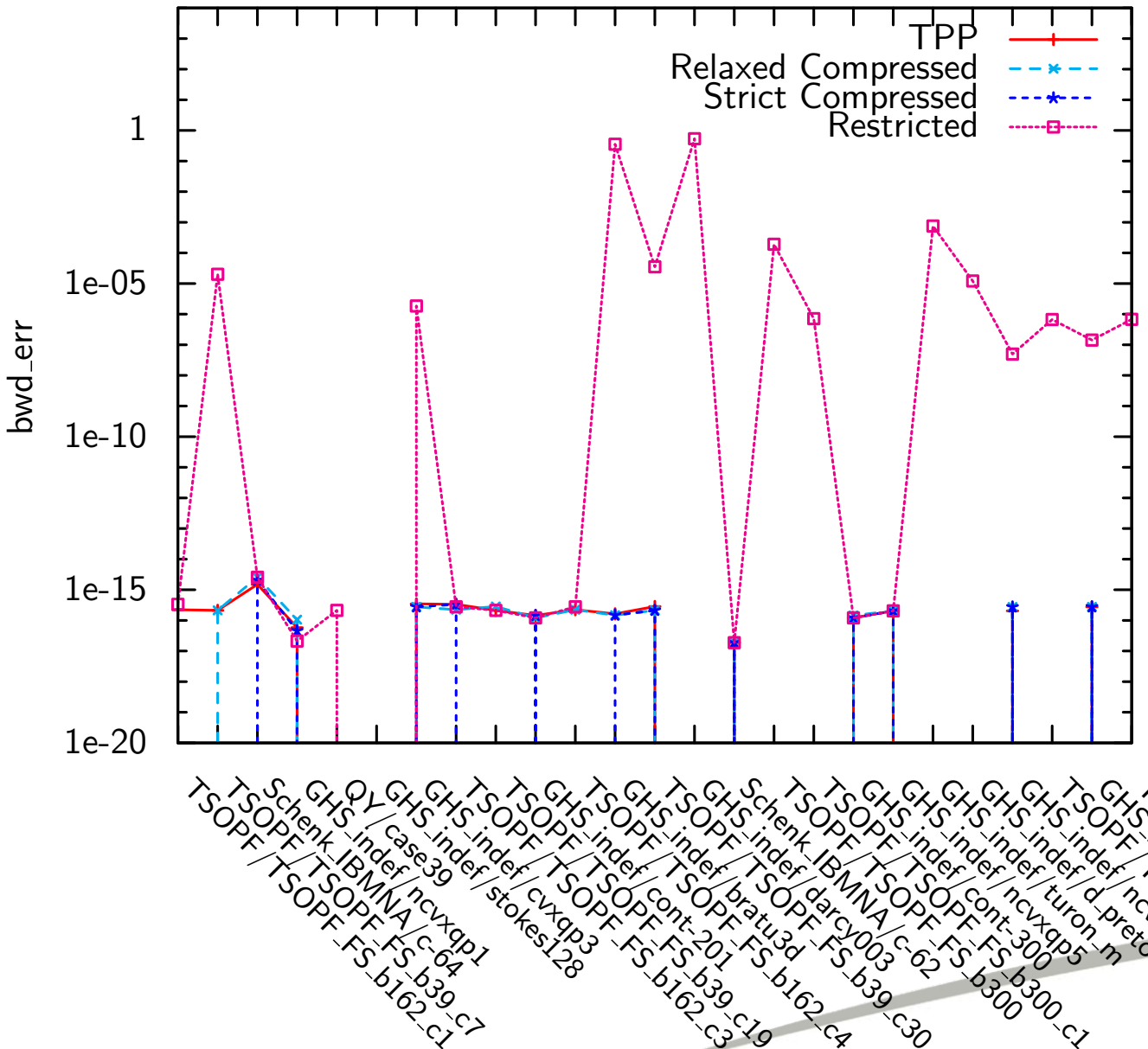


25 difficult problems

- Strict and TPP always good
- Relaxed better than restricted

Science & Technology Facilities Council

Compressed Pivoting: Numerical Stability

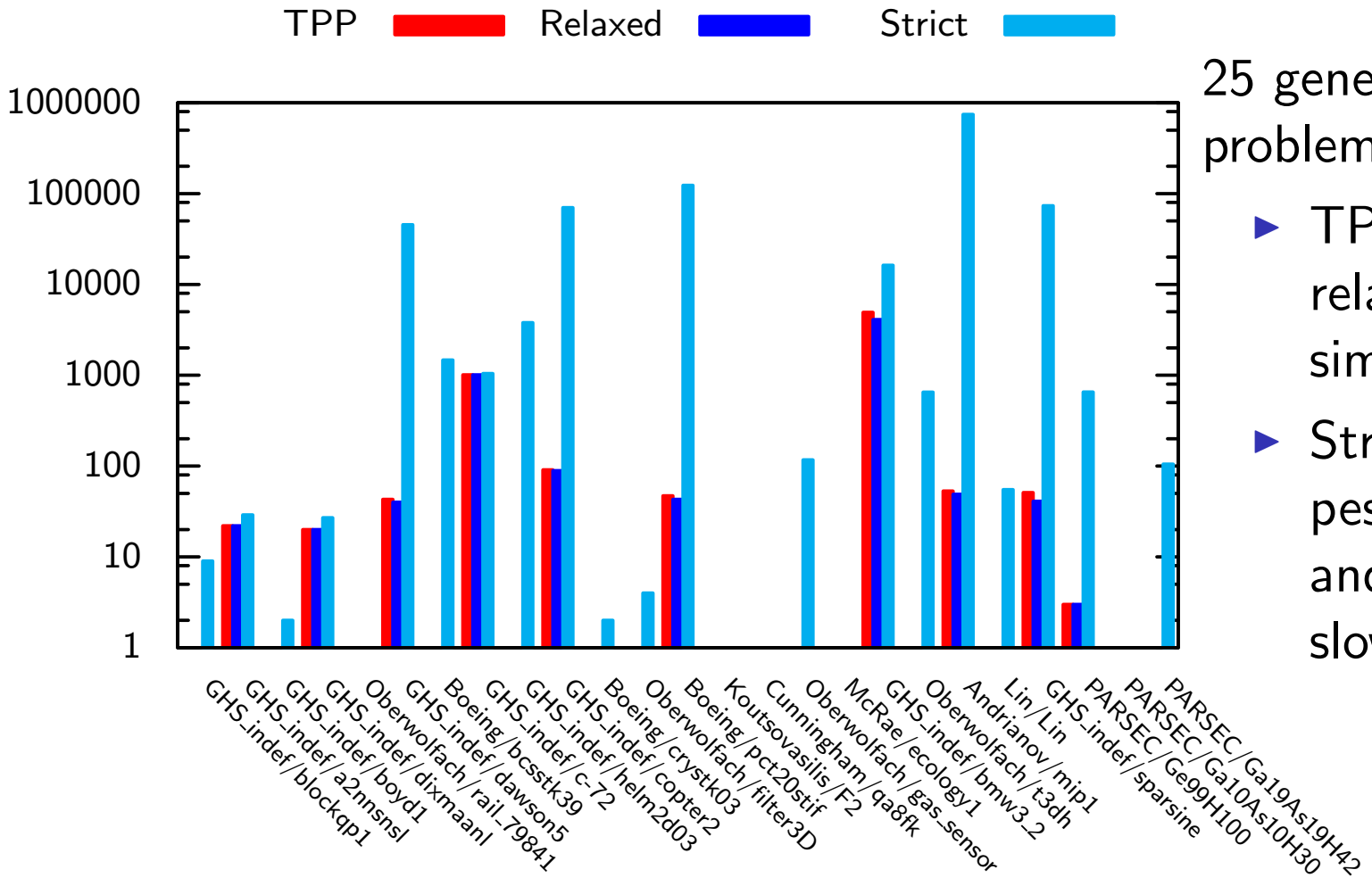


25 difficult problems

- Strict and TPP always good
- Relaxed better than restricted
- Matching-based ordering helps

Science & Technology Facilities Council

Compressed Pivoting: Delays



25 general problems

- TPP and relaxed very similar
- Strict very pessimistic and hence slow

Science & Technology Facilities Council

Putting it all together

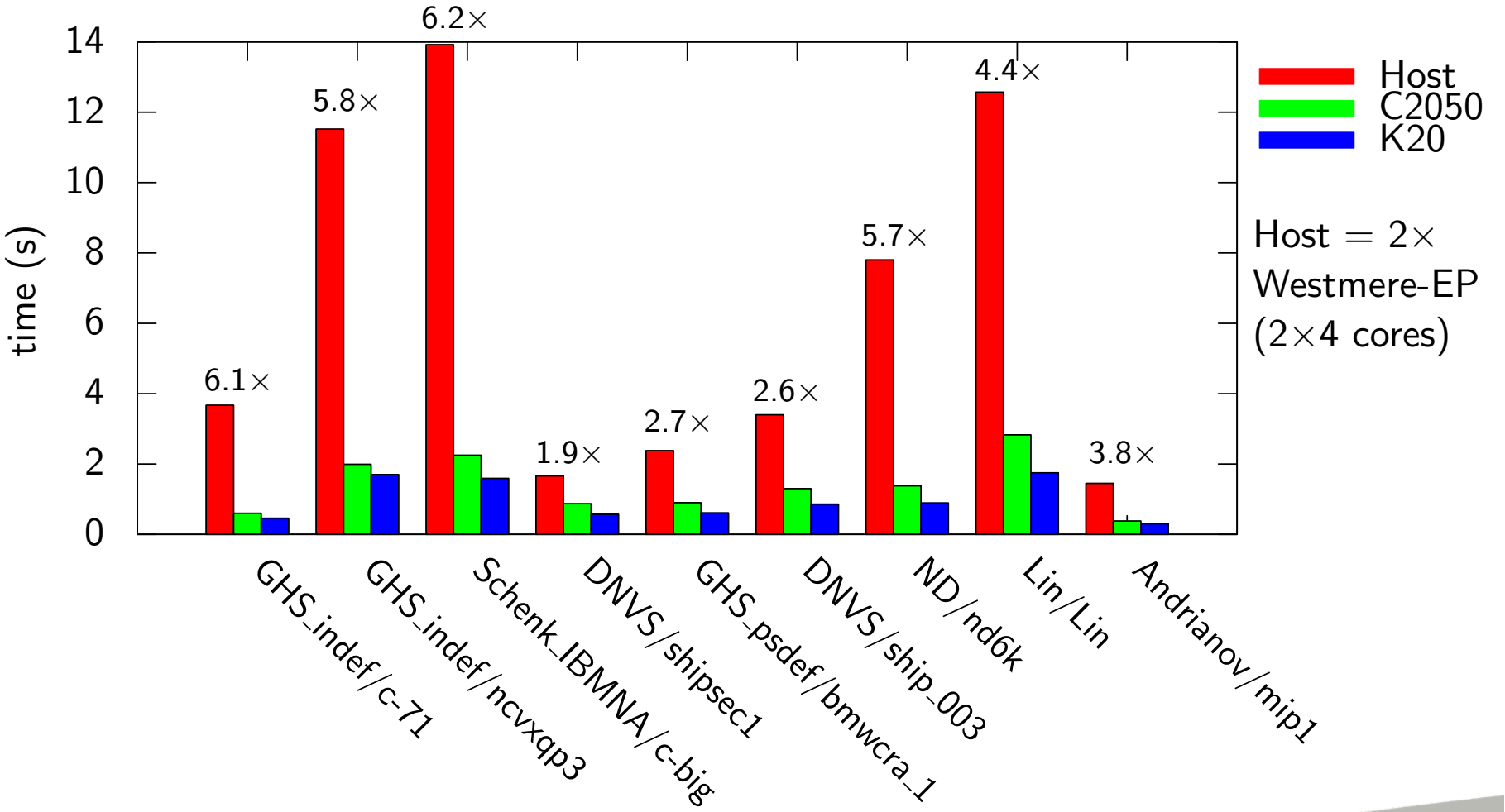
Aims

- Develop GPU and CPU (AVX) implementations
- Try and keep as much in common as possible
- Coarse- and fine-grained task scheduling
- A posteriori pivoting
- Compressed pivoting as fallback

Progress

- GPU solver that uses a posteriori pivoting
- ... but fallback is just to delay failures
- Fine-grained task scheduling in progress
- ... currently undergoing performance optimizations

GPU results



Science & Technology Facilities Council

Thanks for listening!

http://www.numerical.rl.ac.uk/spral

Questions?

A Supplementary slide

Some supplementary text.
(Note numbering of supplementary slides is outside that of normal slides.)