Practical Manycore Pivoting

Jonathan Hogg and Jennifer Scott

STFC Rutherford Appleton Laboratory

30 June 2015
Sparse Days 2015
St Girons, France

* Thanks also to Jeremy Appleyard of NVIDIA

Introduction

Solve: $Ax = b$

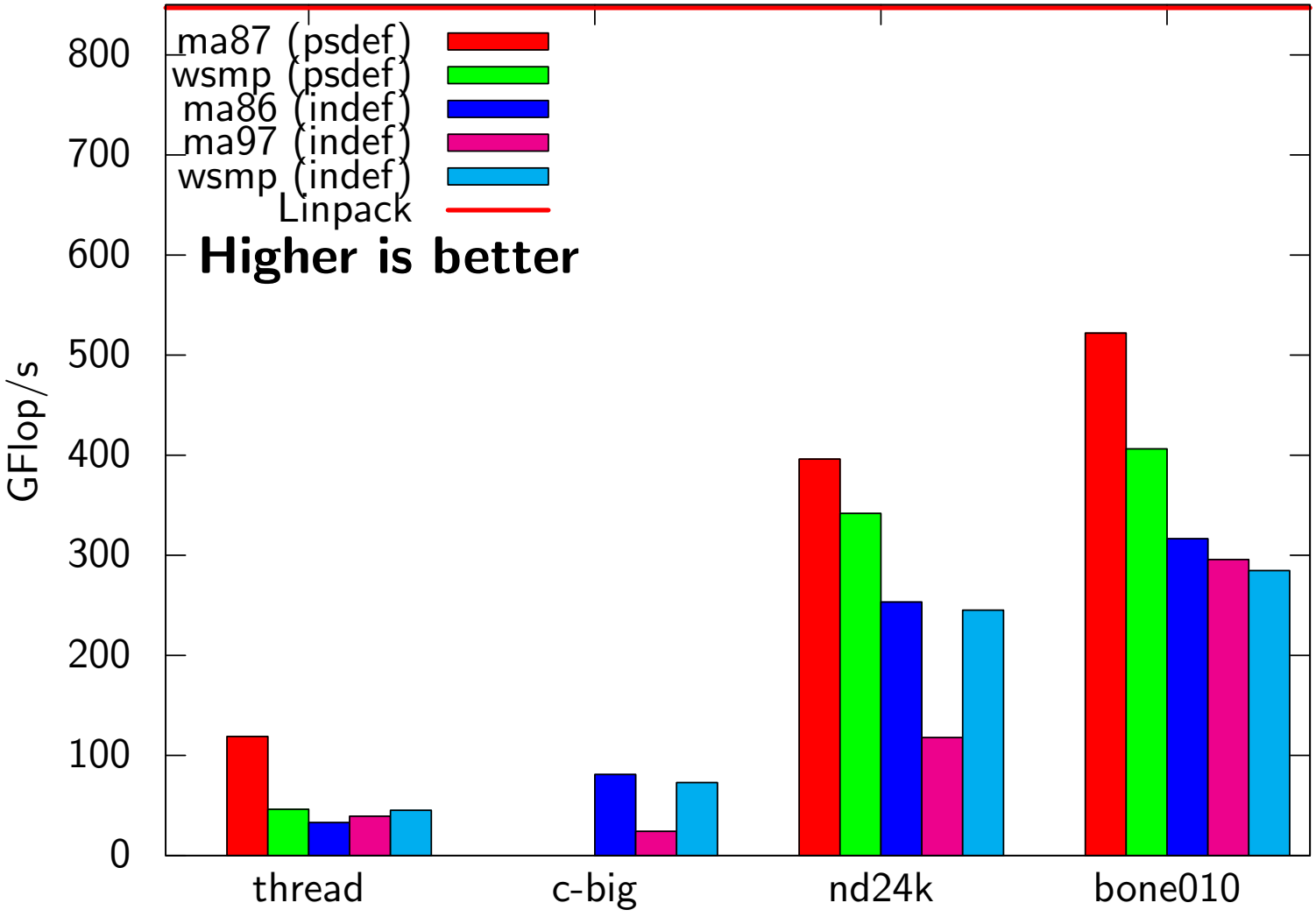
A is:

- Sparse
- Large
- Symmetric
- Indefinite

Algorithm should be:

- Fast
- Accurate
- Numerically Stable
- Bitwise-reproducible?

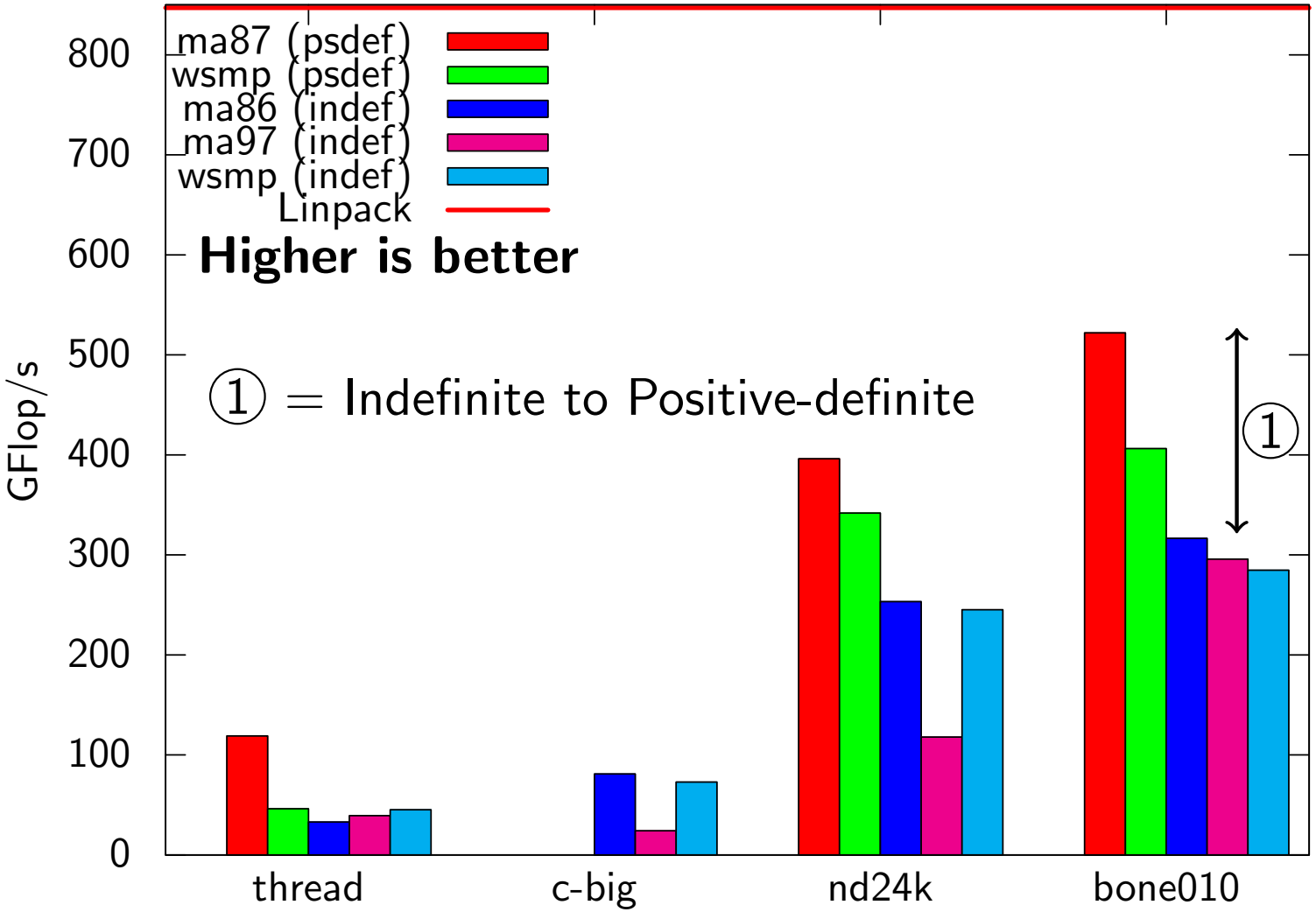
Current Performance Gaps



2×E5-2695 v3 (Haswell-EP) = 28 cores

Science & Technology Facilities Council

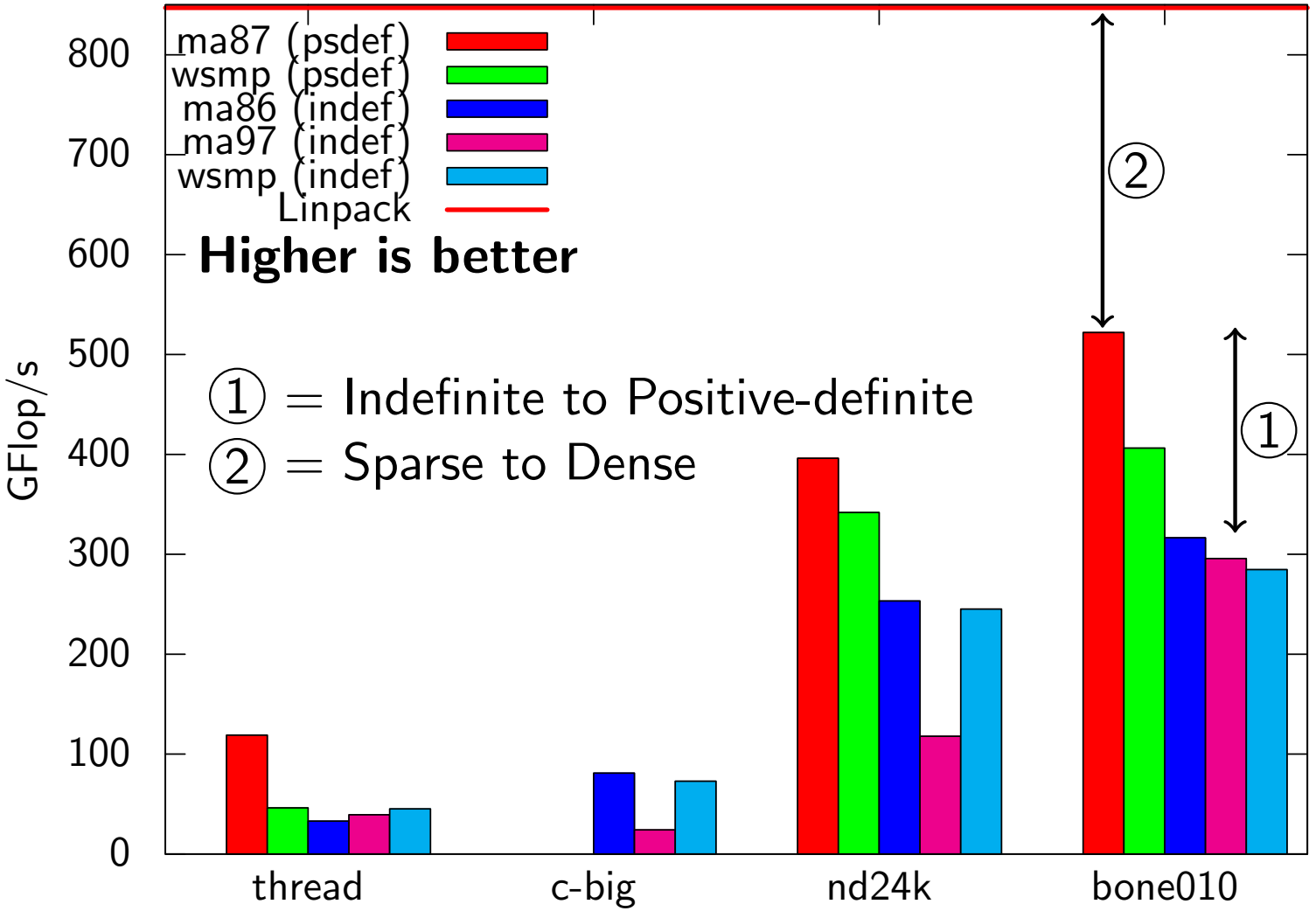
Current Performance Gaps



2×E5-2695 v3 (Haswell-EP) = 28 cores

Science & Technology Facilities Council

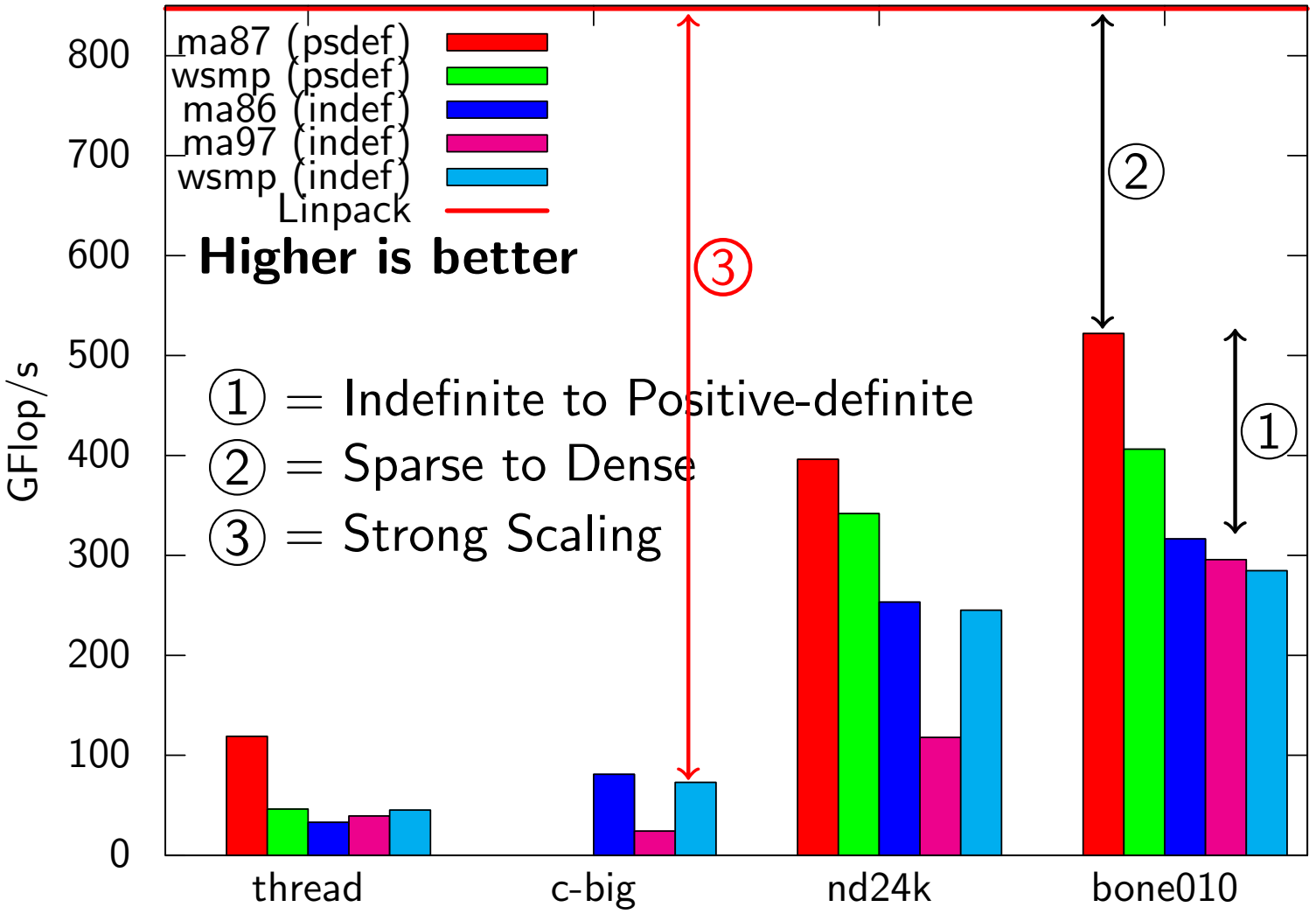
Current Performance Gaps



2×E5-2695 v3 (Haswell-EP) = 28 cores

Science & Technology Facilities Council

Current Performance Gaps



2×E5-2695 v3 (Haswell-EP) = 28 cores

Main focus of this talk

Indefinite to Positive-definite Performance Gap

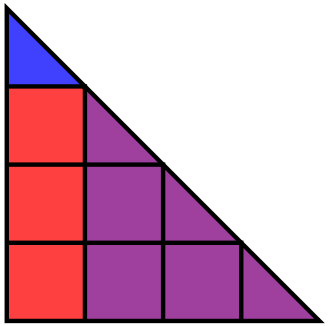
$\Rightarrow$ 20–60% depending on problem

So what's the problem?

Pivoting!

Cholesky pattern:

1. Factor diagonal $\mathbf{A_{jj}} = \mathbf{L_{jj}}\mathbf{L_{jj}}^{-T}$ (Factor)
2. Apply pivot $\mathbf{L_{ij}} = \mathbf{L_{ij}}\mathbf{L_{jj}}^{-T}$ (TRSM)
3. Update uneliminated $\mathbf{A_{ik}} -= \mathbf{L_{ij}}\mathbf{L_{kj}}^{T}$ (GEMM)



Main difference:

- For LDL^T pivoting requires all blocks in column
- Cholesky starts Factor/Apply/Update as soon as ready

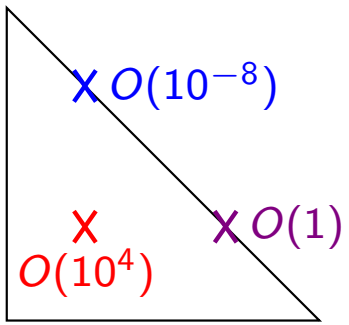
Stability

The problem

- Need to avoid loss of accuracy
- (achieve small backward errors)

Root cause

- "Small" number on diagonal
- "Big" number below diagonal
- Divide one by the other in floating point
- Add to something "small"
- $\Rightarrow$ Lose accuracy
- Resultant large values have large "growth"



Solution

- Don't do this (pivoting)

What's been done before?

Traditional Partial Pivoting (TPP)

- Used in eg MA27, BCSLIB
- "Gold standard" numerical stability
- Not designed with parallelism in mind

What's been done before?

Traditional Partial Pivoting (TPP)

- Used in eg MA27, BCSLIB
- "Gold standard" numerical stability
- Not designed with parallelism in mind

Block Bunch-Kaufman (Richardson '89)

- Used in PARDISO (Schenk '04)
- Just apply dense pivoting in diagonal block
- Potentially unstable
- Preprocessing (scaling, reordering) to alleviate problem

What's been done before?

Traditional Partial Pivoting (TPP)

- Used in eg MA27, BCSLIB
- "Gold standard" numerical stability
- Not designed with parallelism in mind

Block Bunch-Kaufman (Richardson '89)

- Used in PARDISO (Schenk '04)
- Just apply dense pivoting in diagonal block
- Potentially unstable
- Preprocessing (scaling, reordering) to alleviate problem

A posteriori pivoting (Kim and Eijkhout '12)

- Apply pivot, then check growth factor
- Can start Factor before entire column is ready
- Can't start updates until pivot test passed

Our solutions

A posteriori pivoting

- Kim and Eijkhout beat us to publishing the idea.
- Already limited version in SPRAL/SSIDS.
- But we want to take it further.

Our solutions

A posteriori pivoting

- Kim and Eijkhout beat us to publishing the idea.
- Already limited version in SPRAL/SSIDS.
- But we want to take it further.

...with speculative execution

- Set in Cholesky task-DAG context
- Run updates speculatively
- Runtime system handles backtracking to previous version

Our solutions

A posteriori pivoting

- Kim and Eijkhout beat us to publishing the idea.
- Already limited version in SPRAL/SSIDS.
- But we want to take it further.

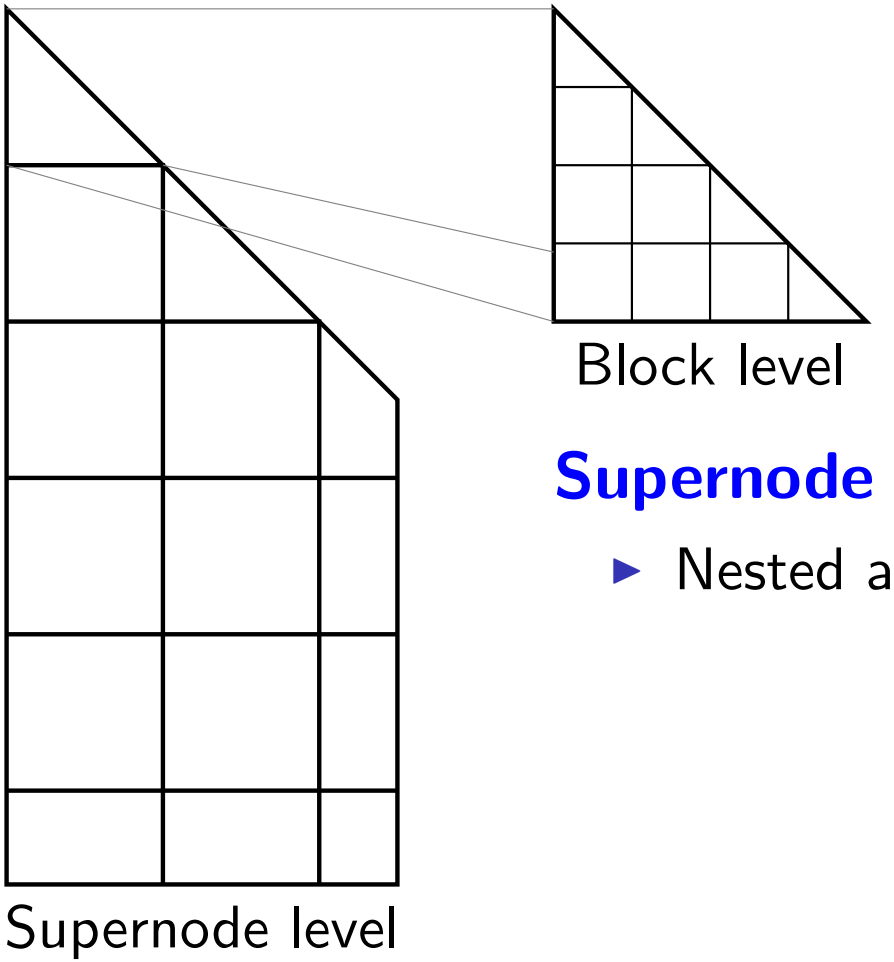
...with speculative execution

- Set in Cholesky task-DAG context
- Run updates speculatively
- Runtime system handles backtracking to previous version

...and fallback

- Detect (subtrees of) matrices with lots of delays
- Handle with new fallback strategy: **compressed TPP**

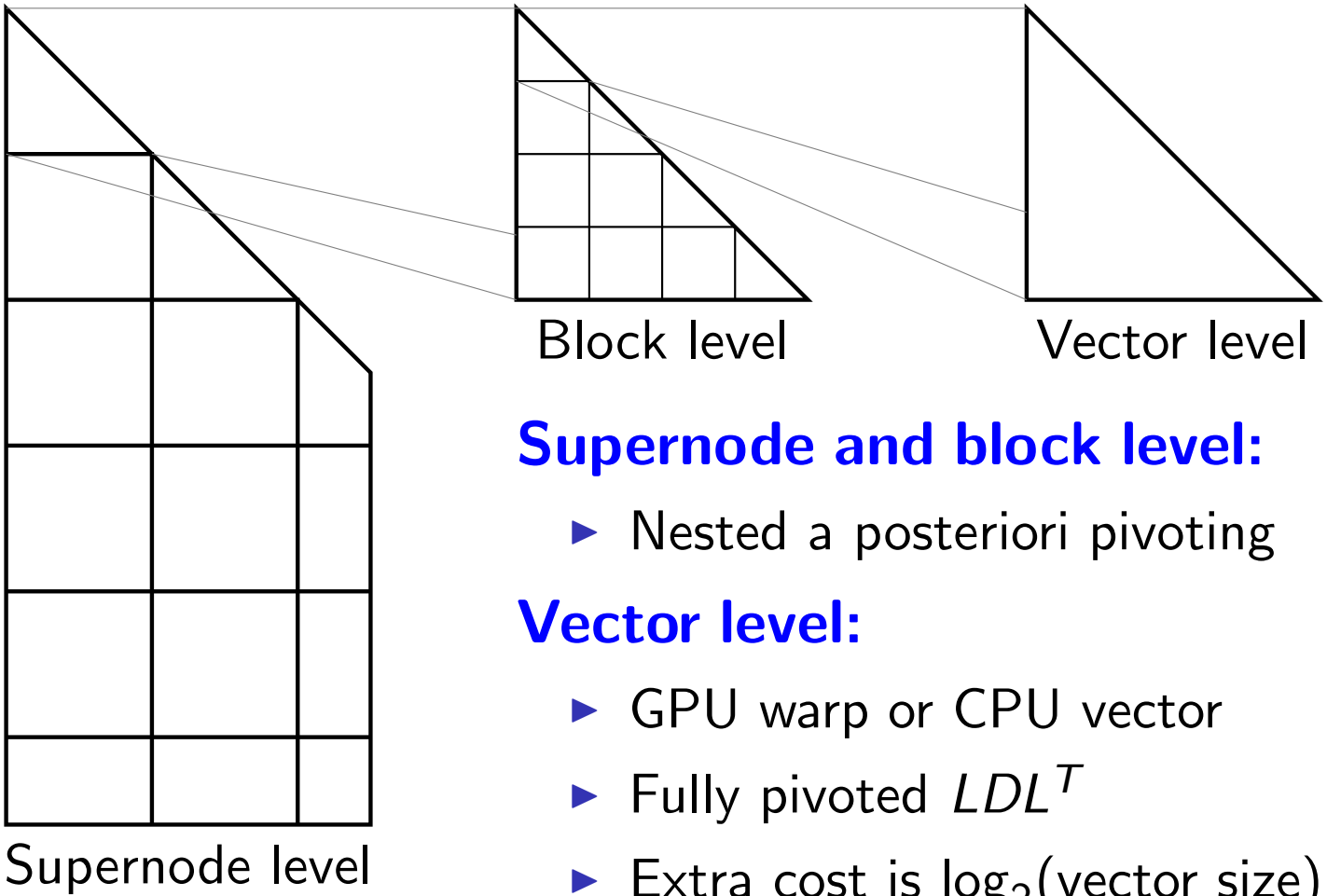
Multi-level approach



Supernode and block level:

- Nested a posteriori pivoting

Multi-level approach



Supernode and block level:

- Nested a posteriori pivoting

Vector level:

- GPU warp or CPU vector
- Fully pivoted LDL^T
- Extra cost is $\log_2$(vector size)
- Equivalent to TPP with $u = 0.25$.

Fail in place?

Traditional approach:

- Column fails $\Rightarrow$ search for column that works
- Swap good pivot with failed pivot and continue
- Lots of data movement. Bad!
- Need to keep collection of failed columns up-to-date.

Fail in place?

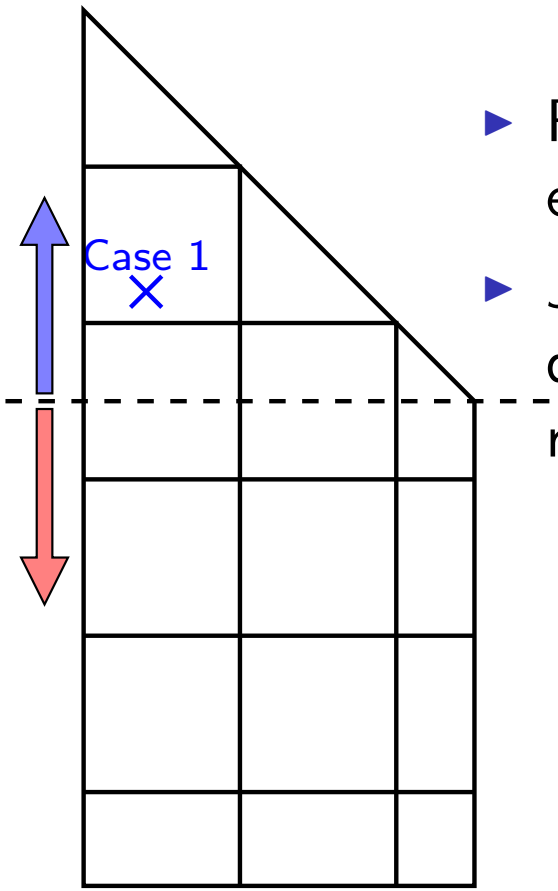
Traditional approach:

- Column fails $\Rightarrow$ search for column that works
- Swap good pivot with failed pivot and continue
- Lots of data movement. Bad!
- Need to keep collection of failed columns up-to-date.

Our approach:

- Fail in place. Leave failed columns where they are.
- Swapping occurs only at end
- Still need to keep failed columns up-to-date
- But fits better with task-based method $\Rightarrow$ no swaps between blocks!

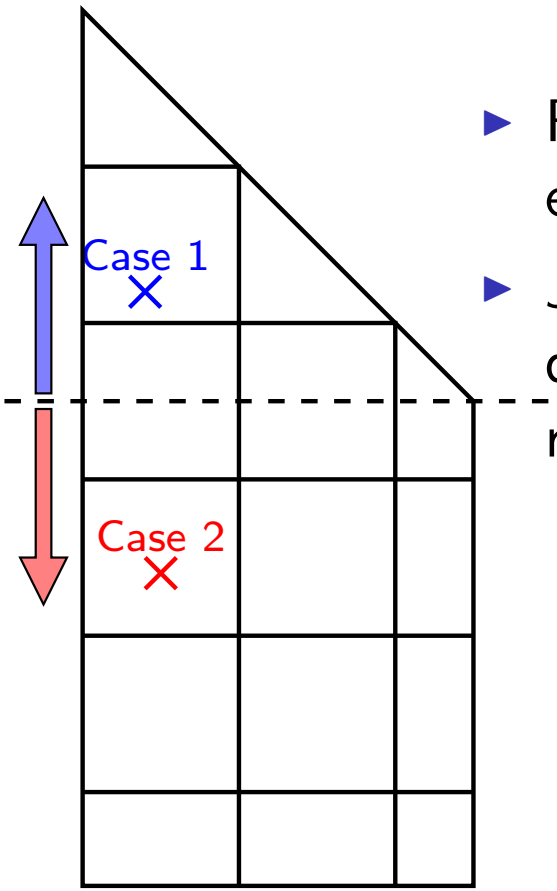
Coping with failure



Case 1:

- Row should be eliminated at this node
- Just swap into a diagonal block and rerun

Coping with failure



Case 1:

- Row should be eliminated at this node
- Just swap into a diagonal block and rerun

Case 2:

- Row can not be eliminated at this node
- Heuristic decision:
  1. Make other swaps and try again
  2. Swap to end and delay

How much swapping?

Terminology

- One iteration = Multiple passes + one round of swapping

Across 25 difficult problems: Matching scaling

- 289 variables eliminated on second or later pass
- 2 variables eliminated on second or later iteration
- (But quite a few delayed to next node)

How much swapping?

Terminology

- One iteration = Multiple passes + one round of swapping

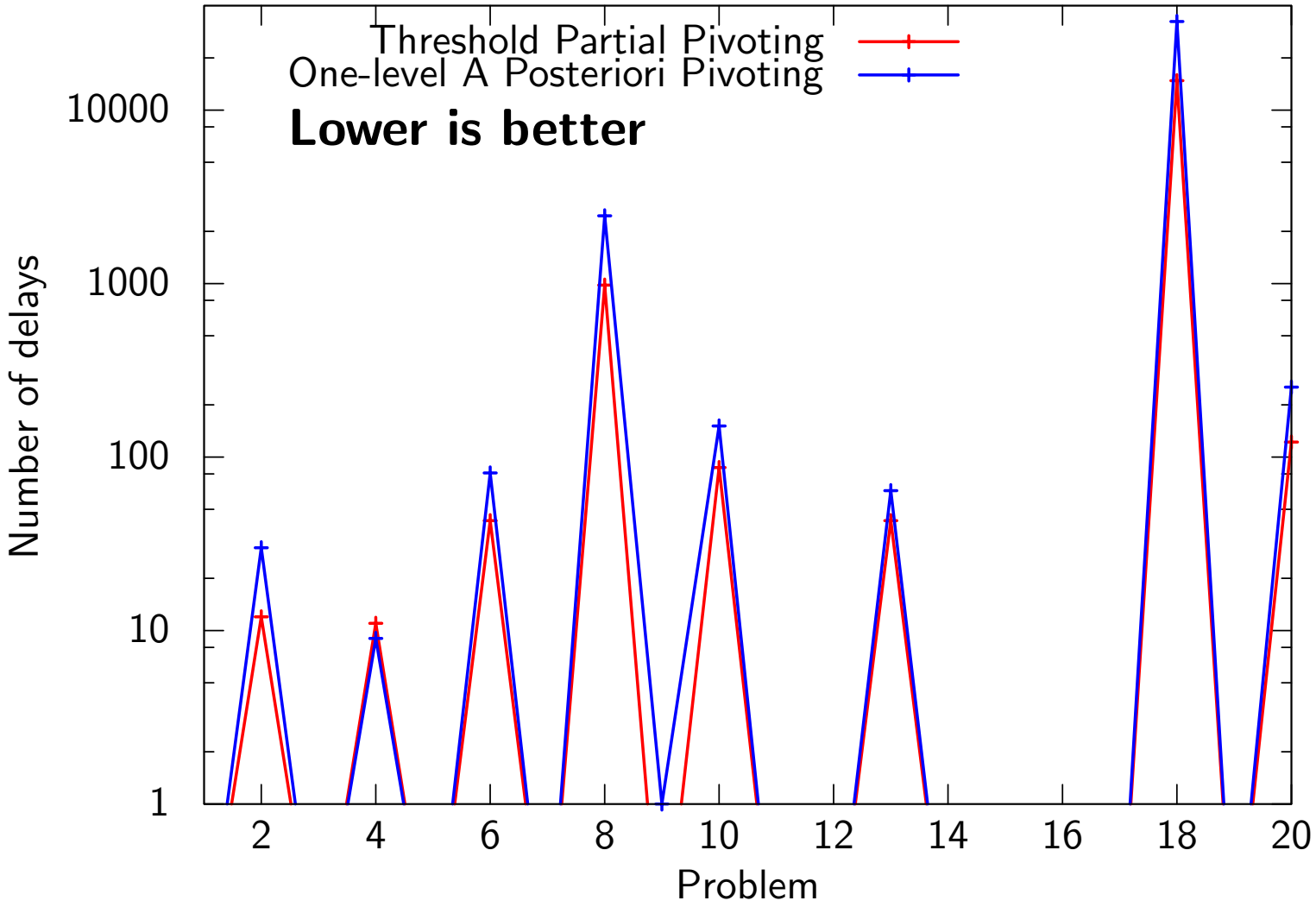
Across 25 difficult problems: Matching scaling

- 289 variables eliminated on second or later pass
- 2 variables eliminated on second or later iteration
- (But quite a few delayed to next node)

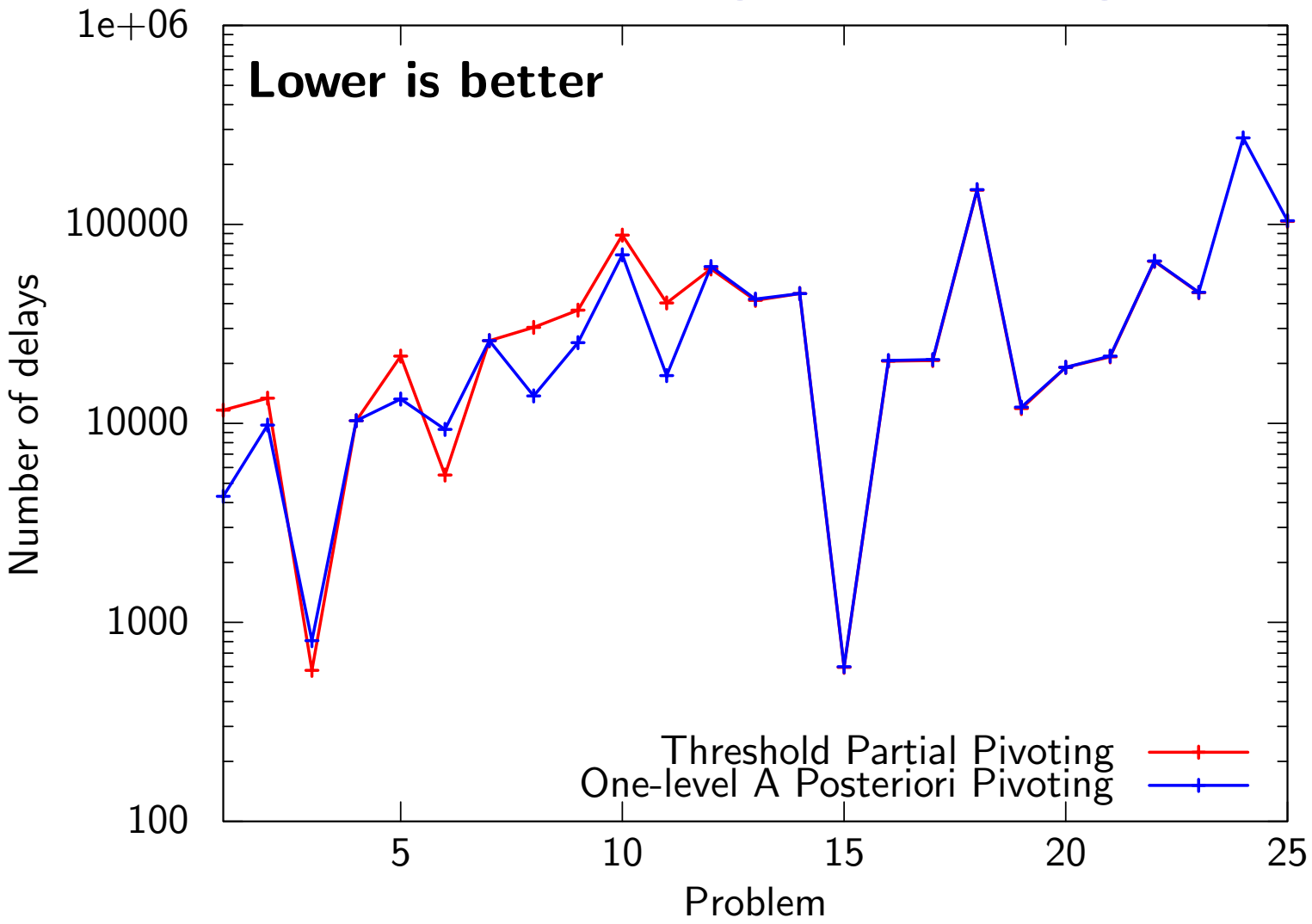
Might as well just delay them?

- Or consolidate and use TPP

Straightforward: no scaling

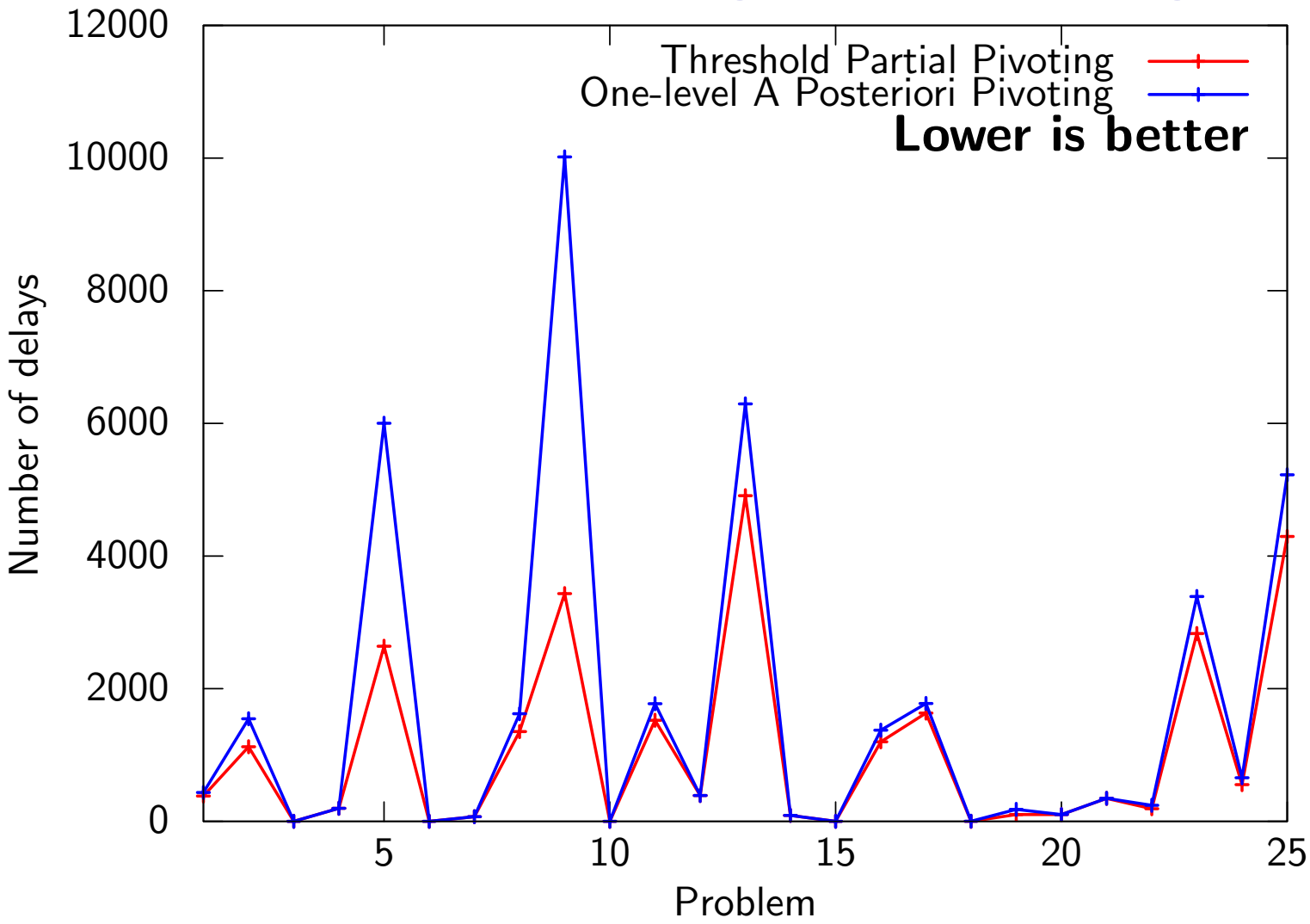


Difficult: Matching Scaling



Science & Technology Facilities Council

Difficult: Matching Ordering



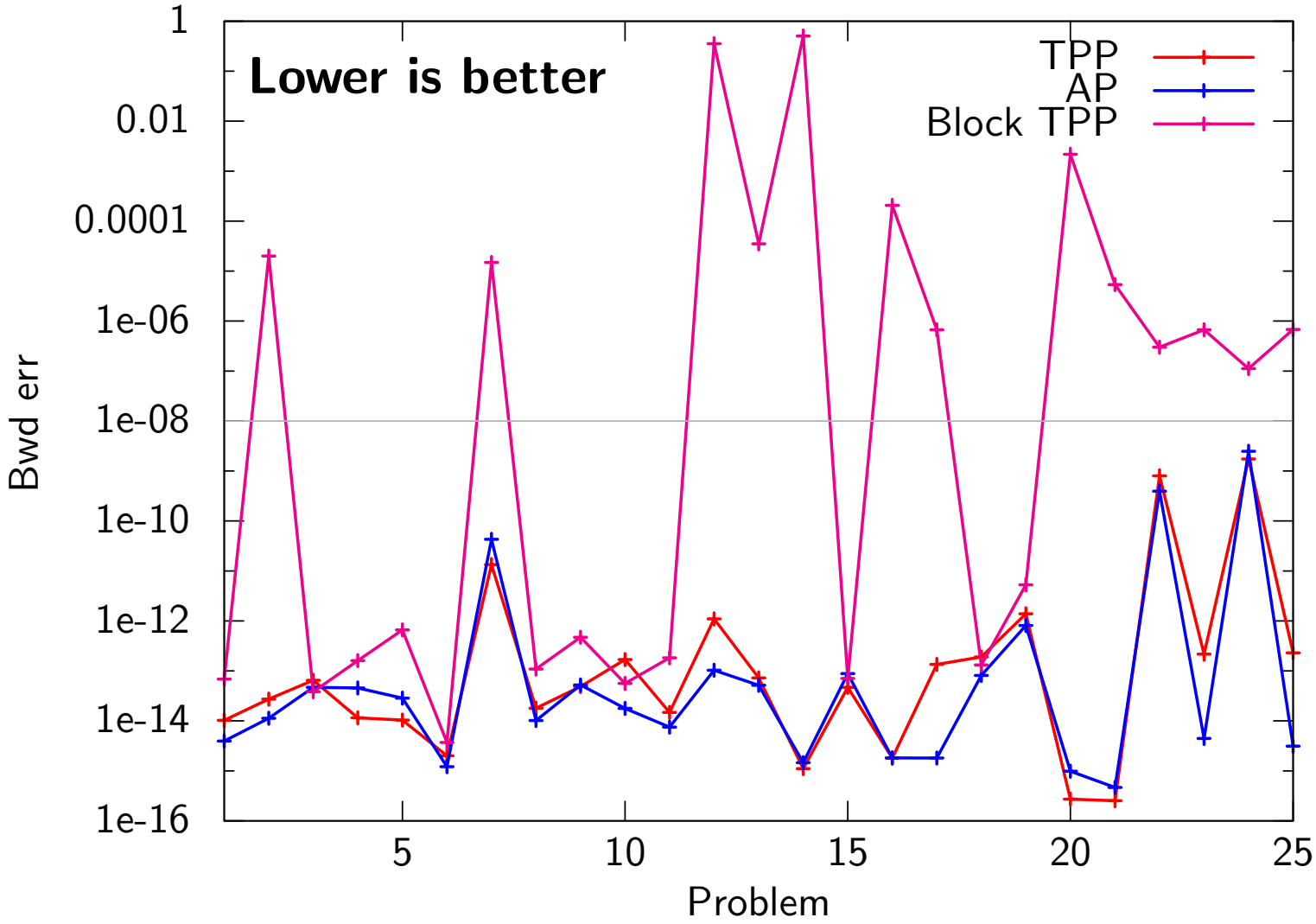
Numerical Accuracy

Consider scaled residual:

$$\frac{\|Ax - b\|_\infty}{\|A\|_\infty \|x\|_\infty + \|b\|_\infty}$$

- Results with best preprocessing (Matching-based ordering)
- Anything $< 10^{-8}$ will *probably* converge using IR

Difficult: Matching-based ordering

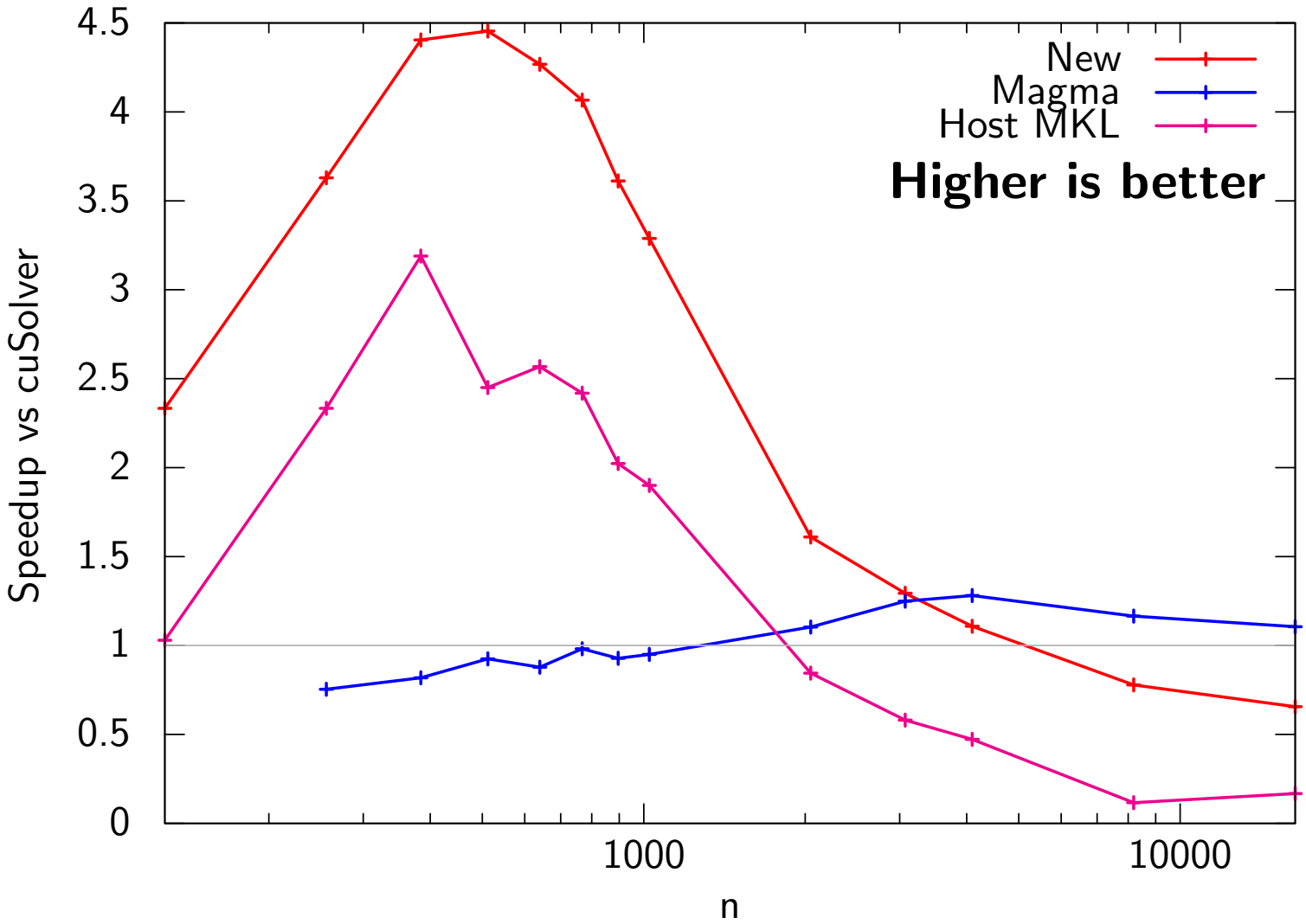


GPU implementation

Still a work in progress...

- Dense Cholesky "prototype" finished
- Dense LDL^T version currently being developed
- Sparse versions pending:
  - "Drop in" to existing solver straightforward
  - Fully task-driven with dynamic parallelism for the future
  - Fully hybrid/multi-GPU code final target

GPU Dense Cholesky: K40



Science & Technology Facilities Council

Conclusions: A Posteriori Pivoting

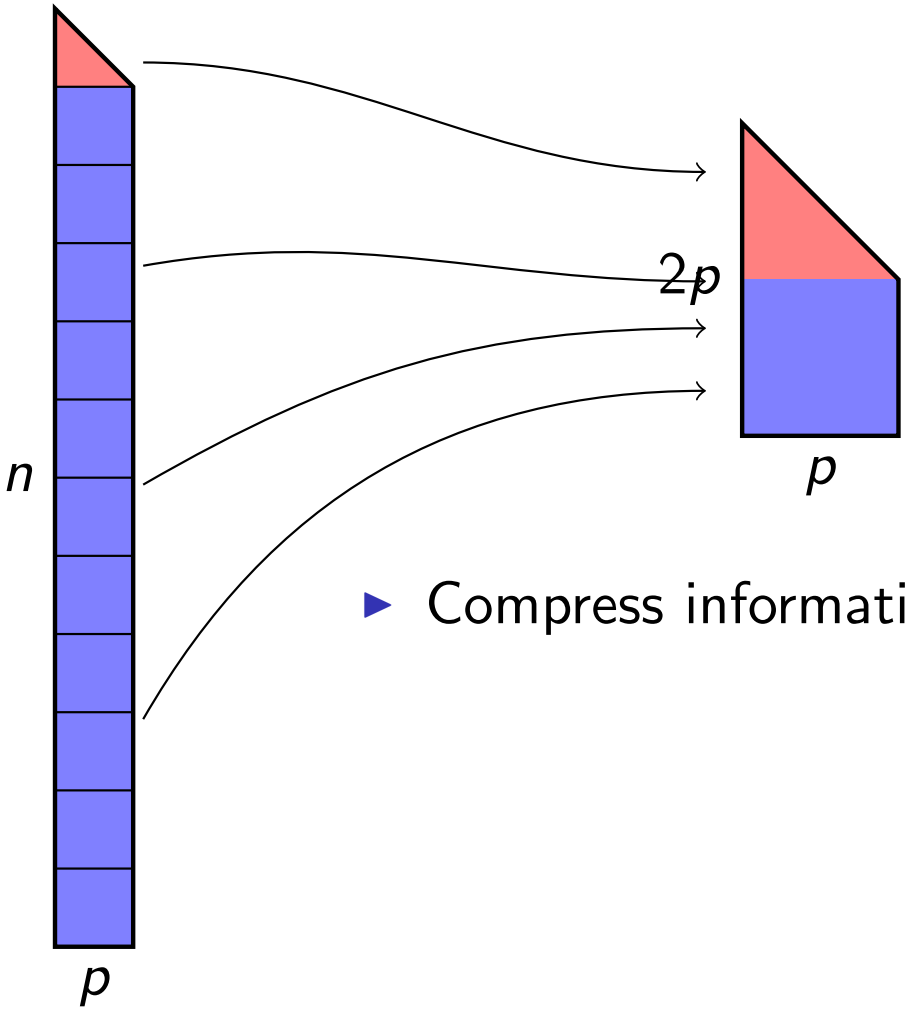
Aim: Symmetric Indefinite to be as cheap as Cholesky

- For many practical problems stability isn't an issue
- ... so check it a posteriori
- Even for numerically difficult problems cost isn't high

Codes will be forthcoming

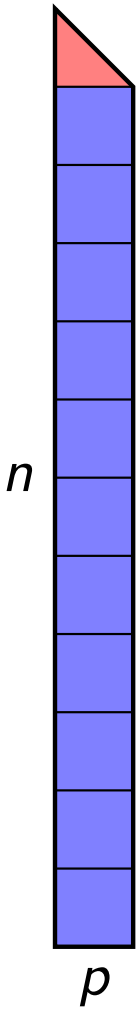
- GPU code takes longer to develop than I would like
- `http://www.numerical.rl.ac.uk/spral`

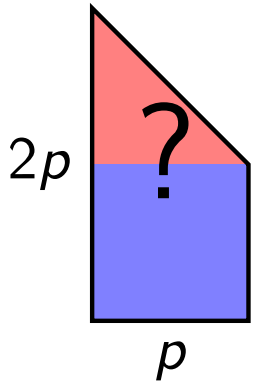
Fallback: Compressed pivoting



- Compress information into small matrix

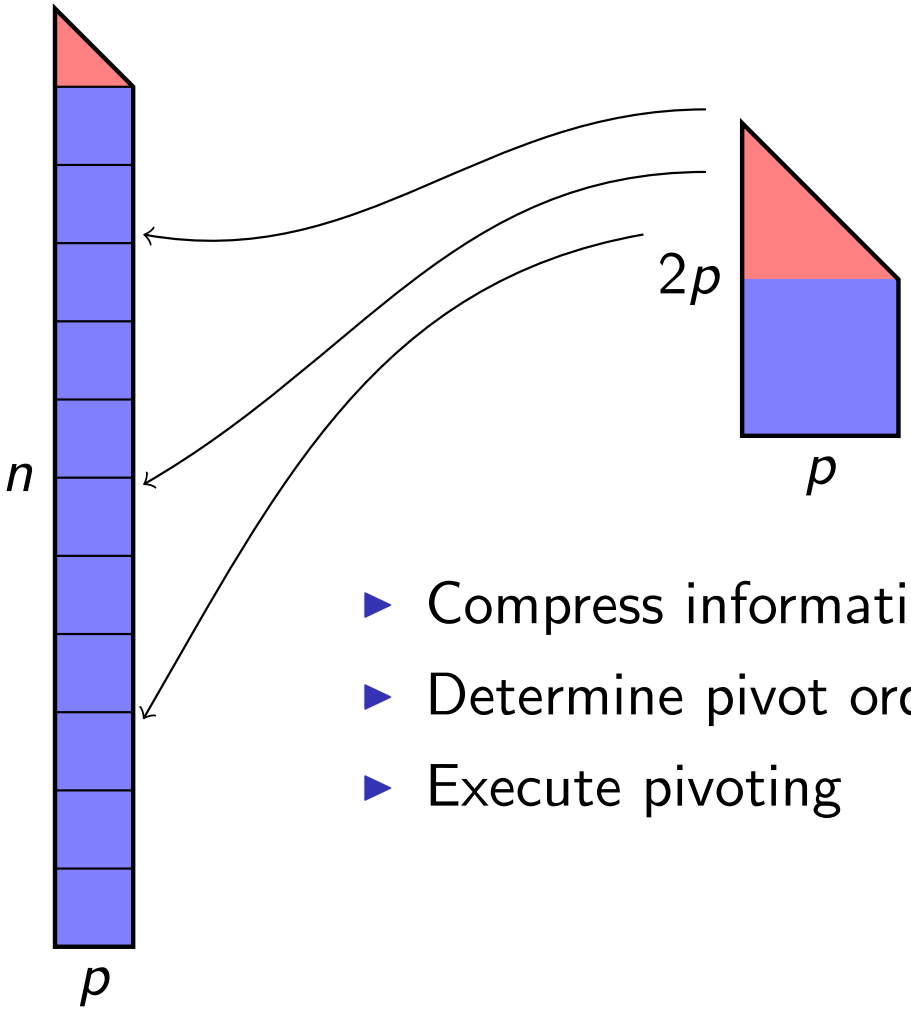
Fallback: Compressed pivoting





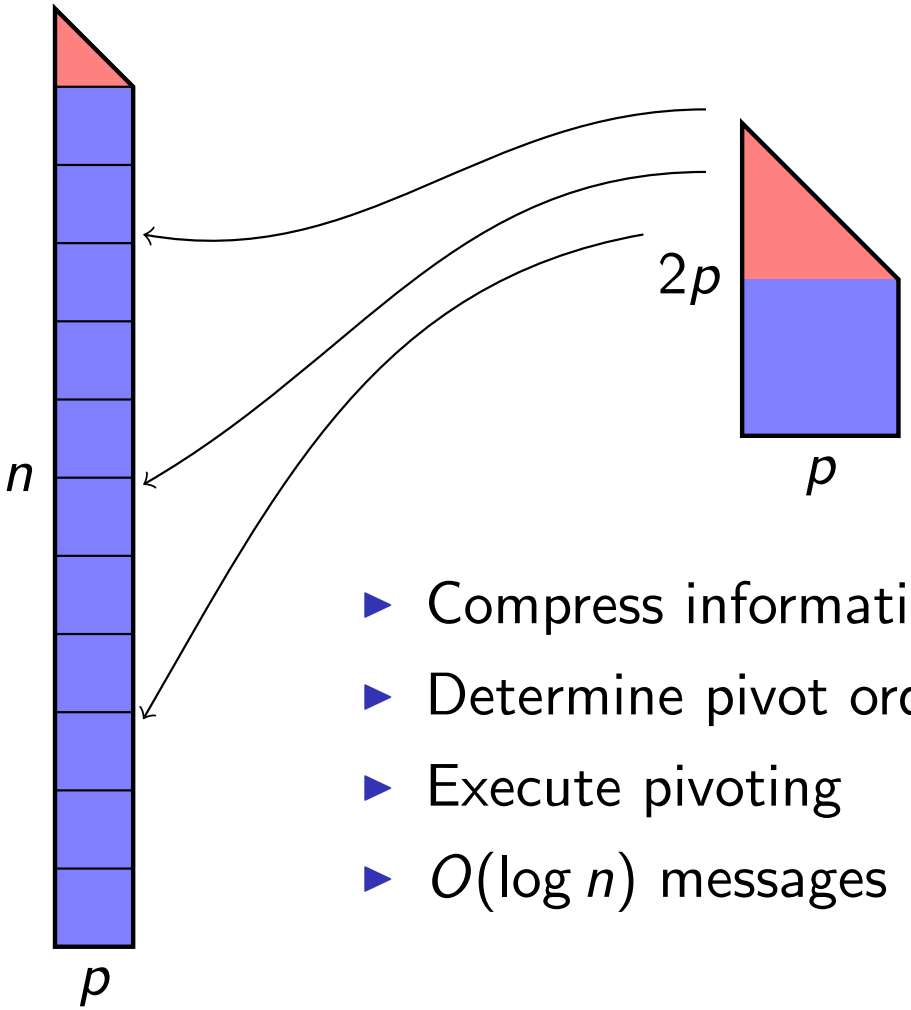
- Compress information into small matrix
- Determine pivot order

Fallback: Compressed pivoting



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

Fallback: Compressed pivoting



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

Strategies

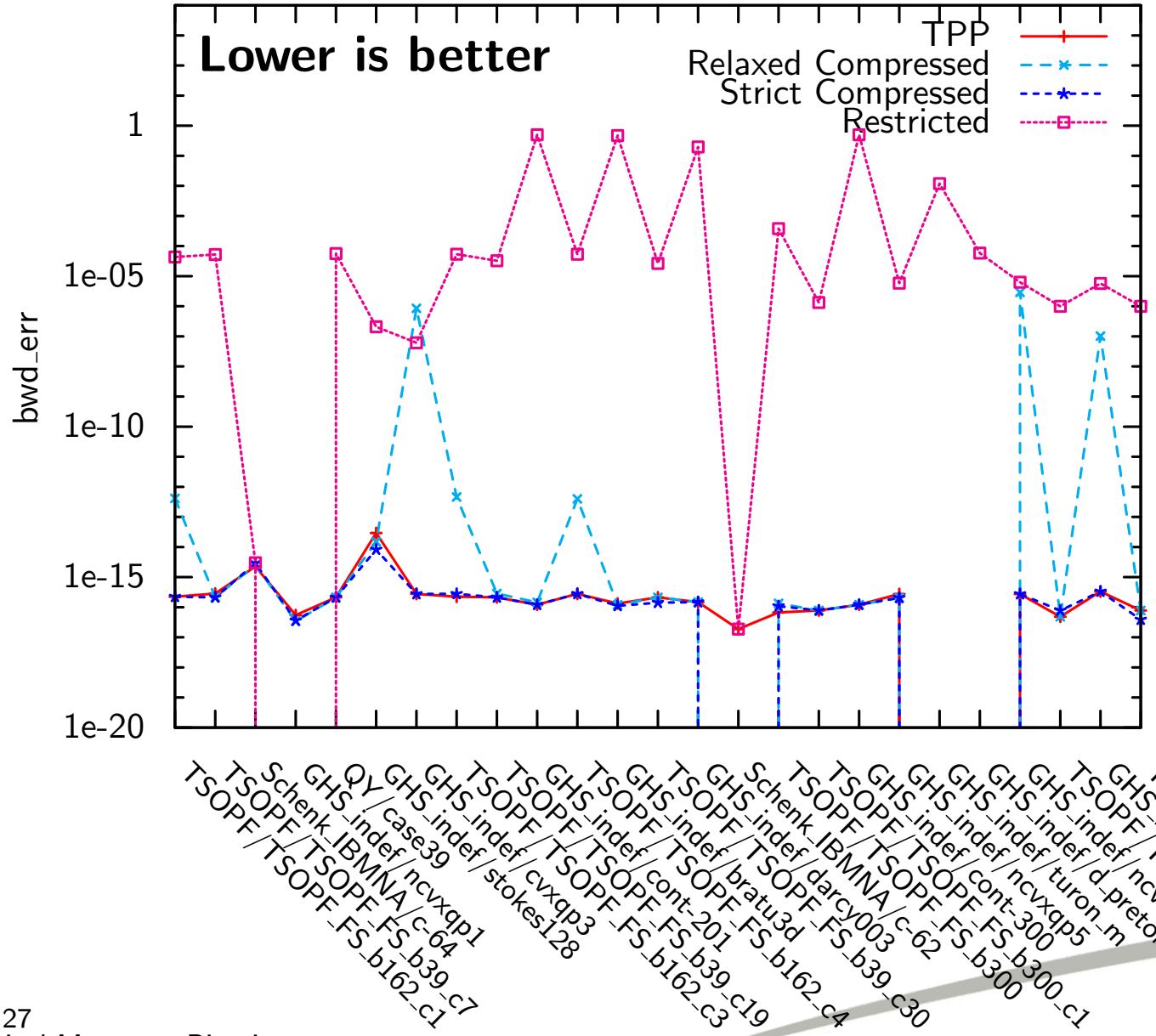
Strict Compressed Pivoting

- Provably numerically stable
- Expect worst case growth
- Very pessimistic $\Rightarrow$ more delays

Relaxed Compressed Pivoting

- Demonstrably unstable on pathological examples
- Stable in practice — see results
- Similar performance to traditional TPP

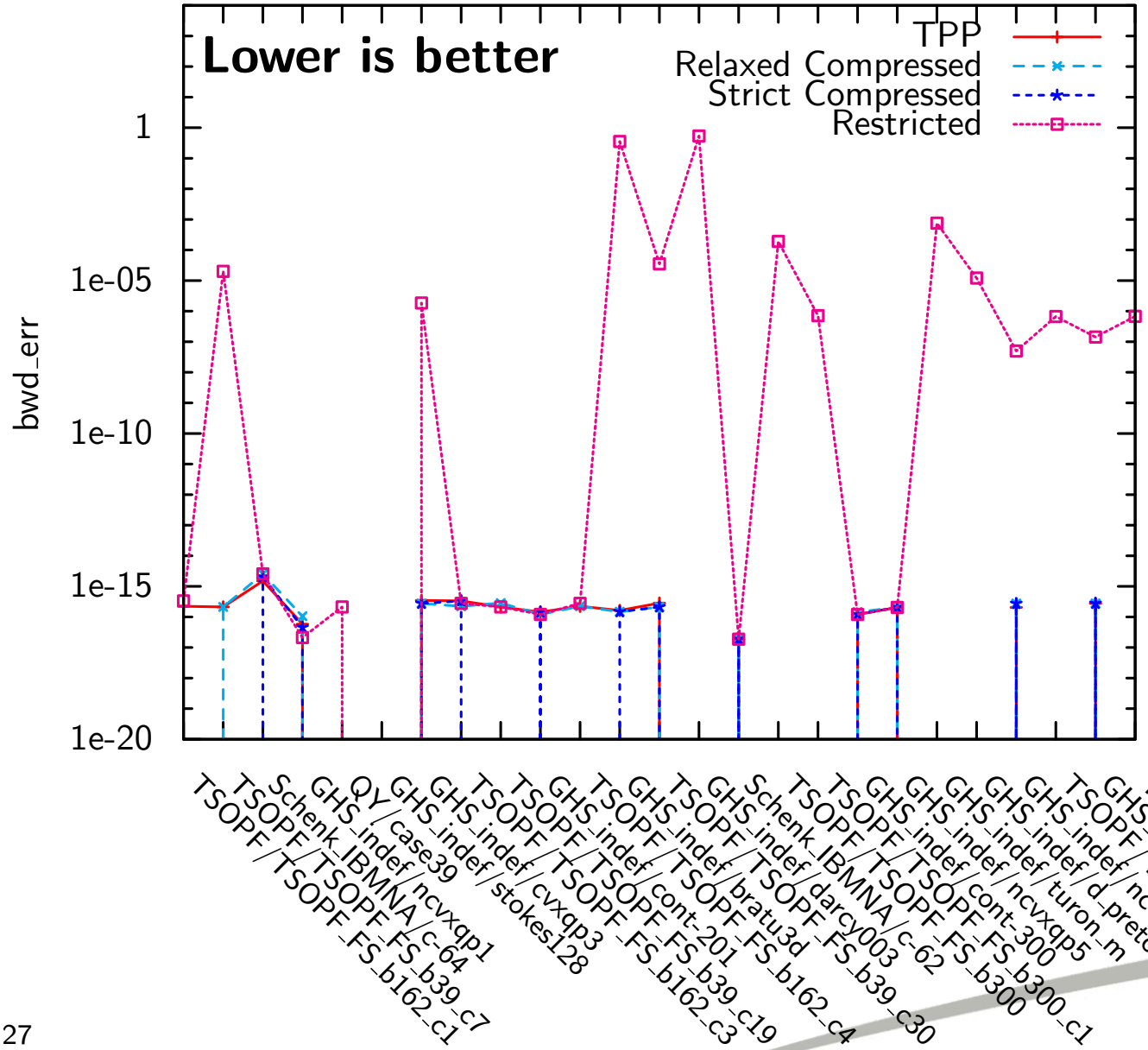
Results: numerical stability



25 difficult problems

- Strict and TPP always good
- Relaxed better than restricted

Results: numerical stability

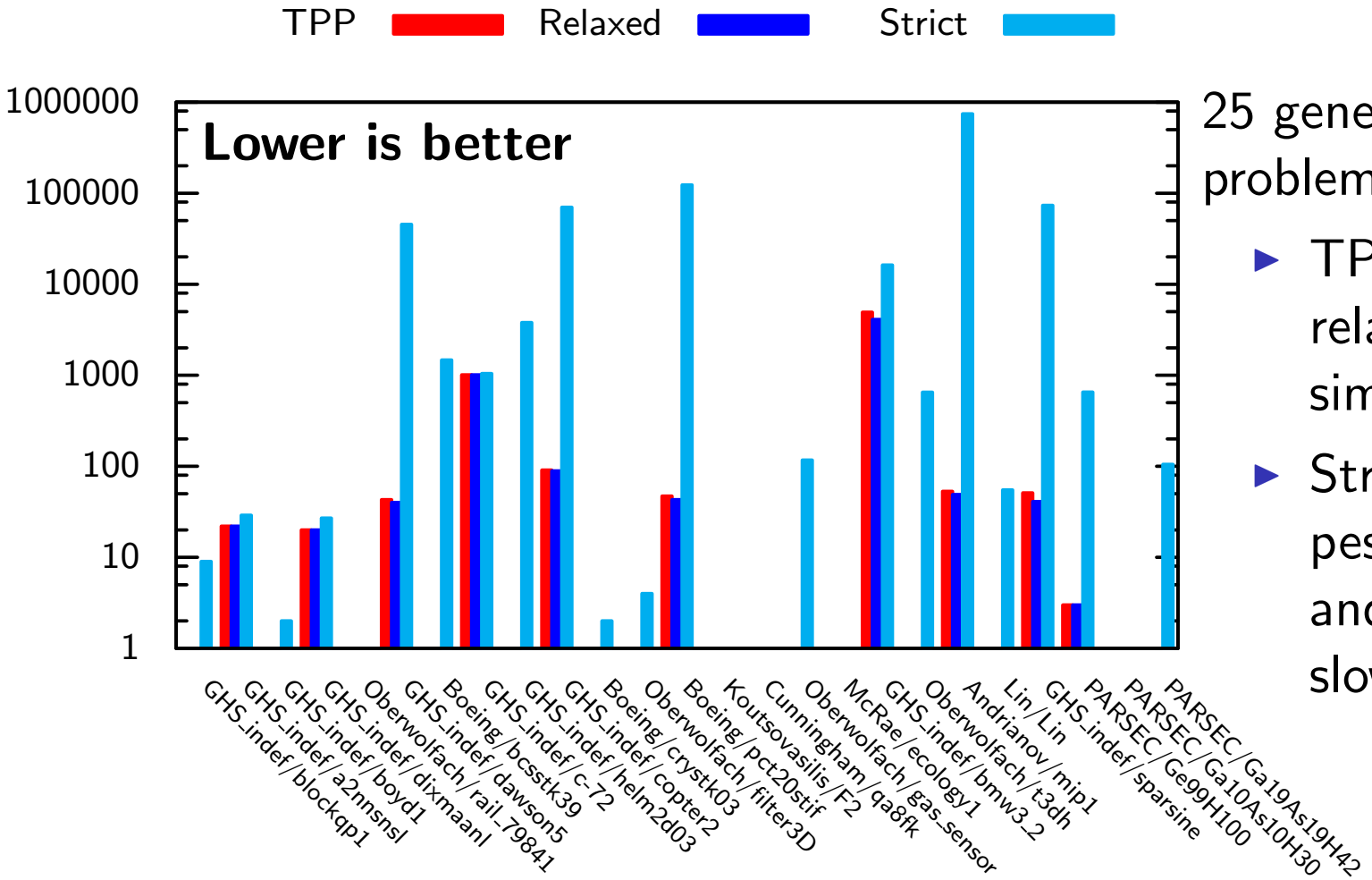


25 difficult problems

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

Science & Technology Facilities Council

Results: delays



25 general problems

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

Science & Technology Facilities Council

Compressed Pivoting: Conclusions

Summary

- CPU compressed pivoting 2+ times faster on large problems
- Restricted pivoting not good enough for all problems
- Strict compressed pivoting guarantees backwards stability
- Relaxed compressed pivoting works well and cheaper in practice
- Good fallback method for when a posteriori pivoting encounters lots of delays

Thanks for listening!

Questions?

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

Tasks

Cholesky-like

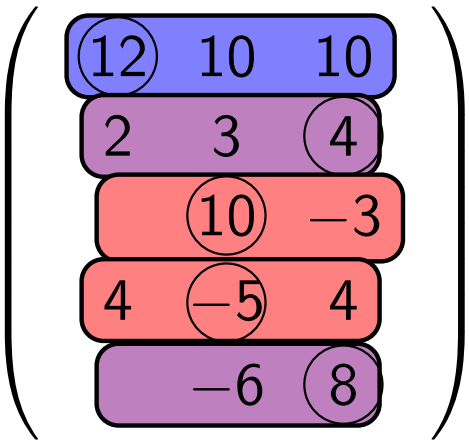
- Factor Diagonal Block
- Apply Block Pivot
- Update to Right

Revisit nodes

- Apply permutation to left
- Apply pivot to left (unelim only) + test?
- Consolidate uneliminated pivots

Strict Compressed Pivoting

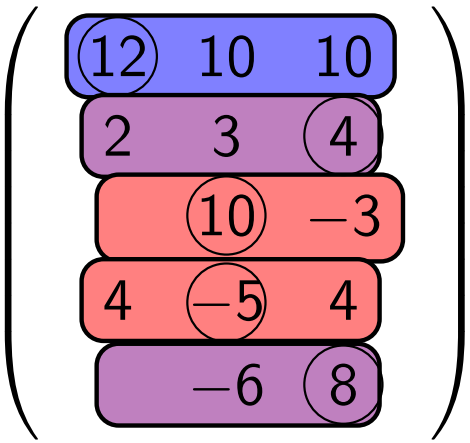
1. Partition rows into sets by column of maximum $|a_{ij}|$



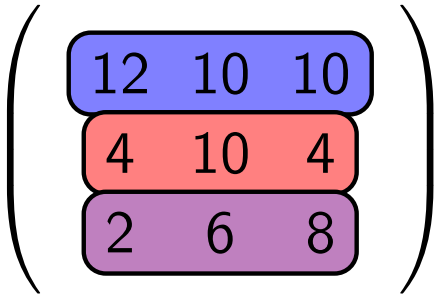
Partitioned rows

Strict Compressed Pivoting

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



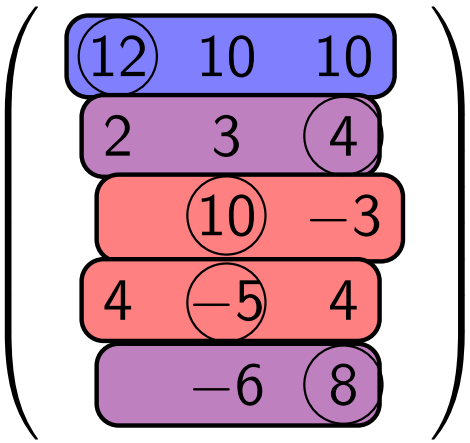
Partitioned rows



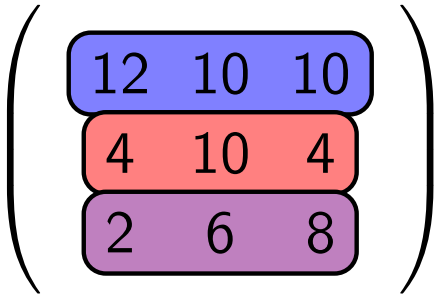
Compressed matrix

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

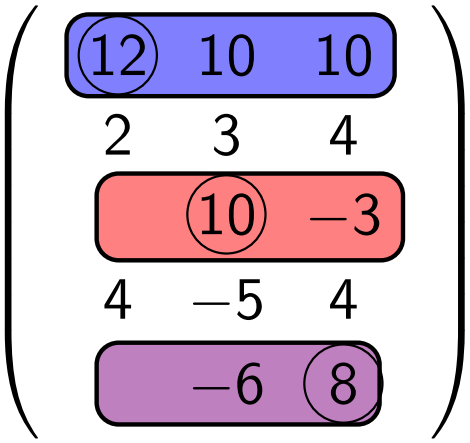


Compressed matrix

- Provably backwards stable
- Sometimes too pessimistic

Relaxed example

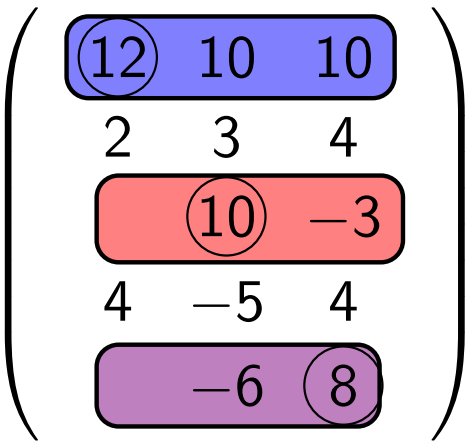
1. For each column, pick a "representative" row: largest $|a_{ij}|$
2. Apply standard threshold partial pivoting.



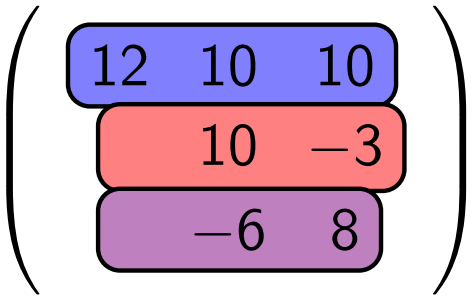
Partitioned rows

Relaxed example

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



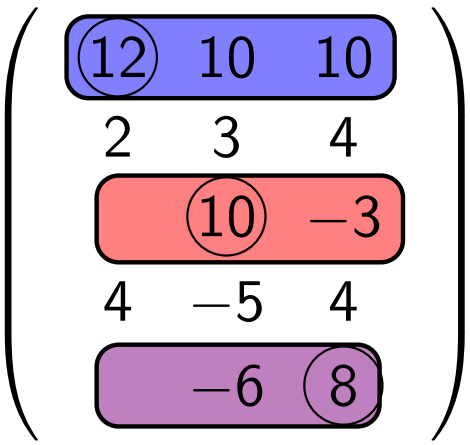
Partitioned rows



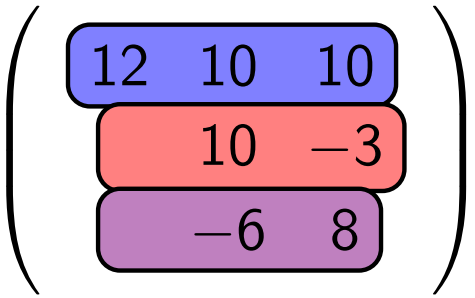
Compressed matrix

Relaxed example

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)