Science & Technology Facilities Council

Sparse Communication Avoiding Pivoting

Jonathan Hogg
Jennifer Scott

STFC Rutherford Appleton Laboratory

25th Biennial Numerical Analysis Conference
June 2013

Communication avoiding pivoting: why?

Now:

- K10 GPU has 16,384 threads for 1,536 "cores" (or 48 warps)
- Xeon Phi has 240 threads for 60 cores
- Typical workstation 32 threads for 16 cores

Communication avoiding pivoting: why?

Now:

- K10 GPU has 16,384 threads for 1,536 "cores" (or 48 warps)
- Xeon Phi has 240 threads for 60 cores
- Typical workstation 32 threads for 16 cores

Future:

- Exascale about 10,000,000,000 (10 billion) threads
- More, less powerful, lower clocked cores
- Multiple threads per core to hide latencies

Communication avoiding pivoting: why?

Now:

- K10 GPU has 16,384 threads for 1,536 “cores” (or 48 warps)
- Xeon Phi has 240 threads for 60 cores
- Typical workstation 32 threads for 16 cores

Future:

- Exascale about 10,000,000,000 (10 billion) threads
- More, less powerful, lower clocked cores
- Multiple threads per core to hide latencies

More cores = More communication

Communication isn't getting (that much) faster

Sparse direct solvers

Solve:

$$Ax = b$$

Where A is

- Large
- Sparse

and for this talk

- Symmetric

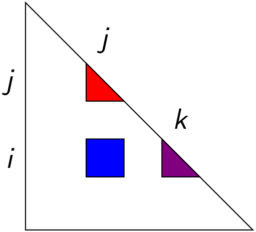
Using the factorization

$$A = LDL^T$$

LDL^T factorization

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 $A_{ik} = A_{ik} - L_{ij} D_{jj} {L_{kj}}^T$



Threshold Partial Pivoting

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

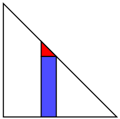
Threshold Partial Pivoting

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

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



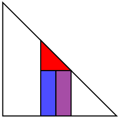
Threshold Partial Pivoting

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

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

Alternatives

Various a priori treatments to reduce/eliminate need for pivoting:

- Scaling. “Normalize” entries of A

Alternatives

Various a priori treatments to reduce/eliminate need for pivoting:

- Scaling. “Normalize” entries of A
- Ordering. Large entries to subdiagonal

Alternatives

Various a priori treatments to reduce/eliminate need for pivoting:

- Scaling. “Normalize” entries of A
- Ordering. Large entries to subdiagonal
- Static pivoting. If a diagonal block is non-singular, add ϵI

Alternatives

Various a priori treatments to reduce/eliminate need for pivoting:

- Scaling. "Normalize" entries of A
- Ordering. Large entries to subdiagonal
- Static pivoting. If a diagonal block is non-singular, add ϵI
- Use as preconditioner e.g. iterative refinement

Alternatives

Various a priori treatments to reduce/eliminate need for pivoting:

- Scaling. "Normalize" entries of A
- Ordering. Large entries to subdiagonal
- Static pivoting. If a diagonal block is non-singular, add ϵI
- Use as preconditioner e.g. iterative refinement

Science & Technology Facilities Council

Alternatives

Various a priori treatments to reduce/eliminate need for pivoting:

- Scaling. "Normalize" entries of A
- Ordering. Large entries to subdiagonal
- Static pivoting. If a diagonal block is non-singular, add ϵI
- Use as preconditioner e.g. iterative refinement

A combination of these approaches works for 95% of real matrices.

Alternatives

Various a priori treatments to reduce/eliminate need for pivoting:

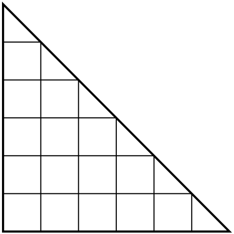
- Scaling. "Normalize" entries of A
- Ordering. Large entries to subdiagonal
- Static pivoting. If a diagonal block is non-singular, add ϵI
- Use as preconditioner e.g. iterative refinement

A combination of these approaches works for 95% of real matrices.

For the other 5% we need pivoting!

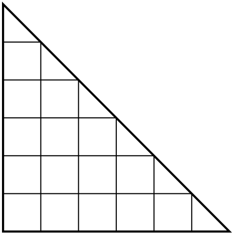
Parallel decomposition

- Apply 2D data decomposition



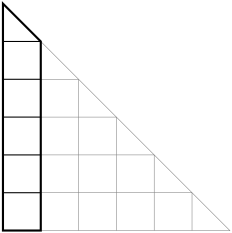
Parallel decomposition

- Apply 2D data decomposition
- Update step parallelizes nicely as per _gemm

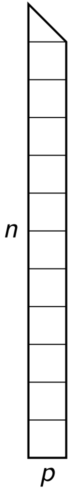


Parallel decomposition

- Apply 2D data decomposition
- Update step parallelizes nicely as per `_gemm`
- For pivoting equivalent to 1D on tall skinny matrix



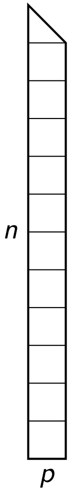
Parallel Variants



Various options:

- Restricted pivoting: only pivot within diagonal block
- Assume all pivots are valid, check L maxima a posteriori
- Traditional TPP

Parallel Variants



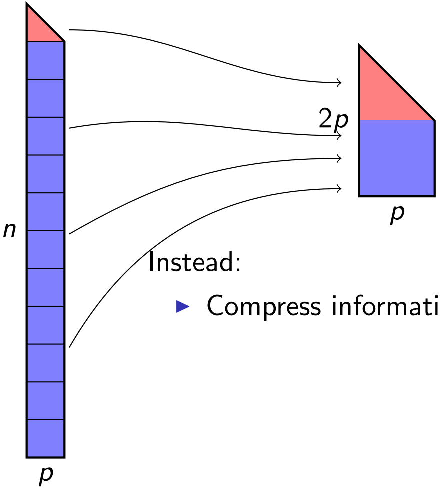
Various options:

- Restricted pivoting: only pivot within diagonal block
- Assume all pivots are valid, check L maxima a posteriori
- Traditional TPP

Parallel variant TPP:

- **Either** one thread owns the diagonal block
- **or** each thread has its own copy of diagonal block
- Regardless, needs a reduction for every pivot
- $O(p \log n)$ messages

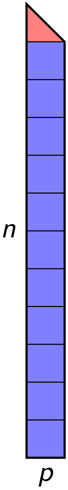
Compressed pivoting

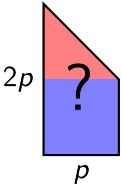


Instead:

- Compress information into small matrix

Compressed pivoting

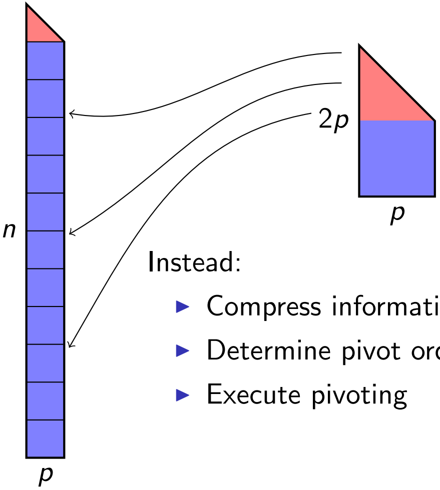




Instead:

- Compress information into small matrix
- Determine pivot order

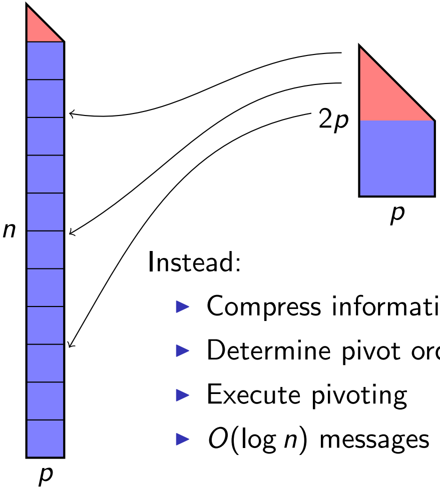
Compressed pivoting



Instead:

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

Compressed pivoting



Instead:

- 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 $|a_{ij}|$

$$\begin{pmatrix} 12 & 10 & 10 \\ 2 & 3 & 4 \\ & 10 & -3 \\ 4 & -5 & 4 \\ & -6 & 8 \end{pmatrix}$$

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 $|a_{ij}|$

$$\begin{pmatrix} 12 & 10 & 10 \\ 2 & 3 & 4 \\ & 10 & -3 \\ 4 & -5 & 4 \\ & -6 & 8 \end{pmatrix}$$

Partitioned rows

$$\begin{pmatrix} 12 & 10 & 10 \\ 4 & 10 & 4 \\ 2 & 6 & 8 \end{pmatrix}$$

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

$$\begin{pmatrix} 12 & 10 & 10 \\ 2 & 3 & 4 \\ & 10 & -3 \\ 4 & -5 & 4 \\ & -6 & 8 \end{pmatrix}$$

Partitioned rows

$$\begin{pmatrix} 12 & 10 & 10 \\ 4 & 10 & 4 \\ 2 & 6 & 8 \end{pmatrix}$$

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.

$$\begin{pmatrix} 12 & 10 & 10 \\ 2 & 3 & 4 \\ & 10 & -3 \\ 4 & -5 & 4 \\ & -6 & 8 \end{pmatrix}$$

Partitioned rows

Relaxed example

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

$$\begin{pmatrix} 12 & 10 & 10 \\ 2 & 3 & 4 \\ & 10 & -3 \\ 4 & -5 & 4 \\ & -6 & 8 \end{pmatrix}$$

Partitioned rows

$$\begin{pmatrix} 12 & 10 & 10 \\ & 10 & -3 \\ & -6 & 8 \end{pmatrix}$$

Compressed matrix

Relaxed example

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

$$\begin{pmatrix} 12 & 10 & 10 \\ 2 & 3 & 4 \\ & 10 & -3 \\ 4 & -5 & 4 \\ & -6 & 8 \end{pmatrix}$$

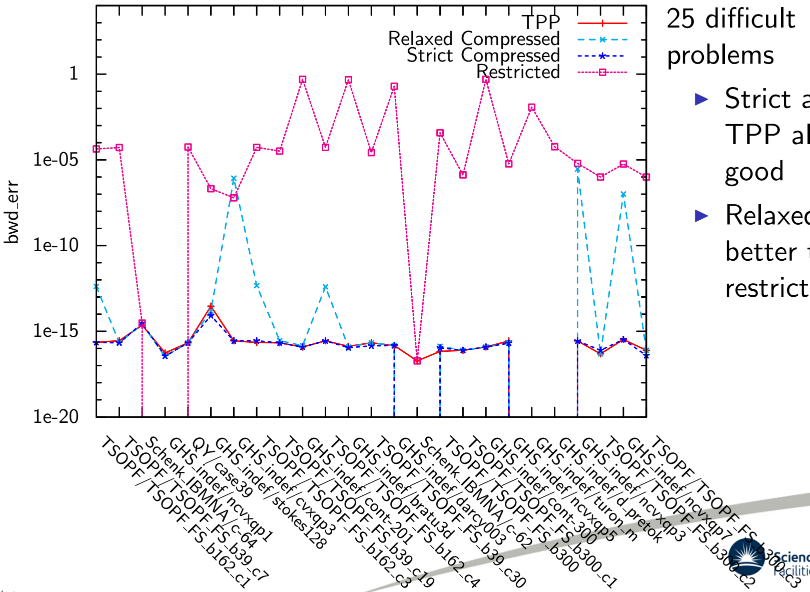
Partitioned rows

$$\begin{pmatrix} 12 & 10 & 10 \\ & 10 & -3 \\ & -6 & 8 \end{pmatrix}$$

Compressed matrix

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

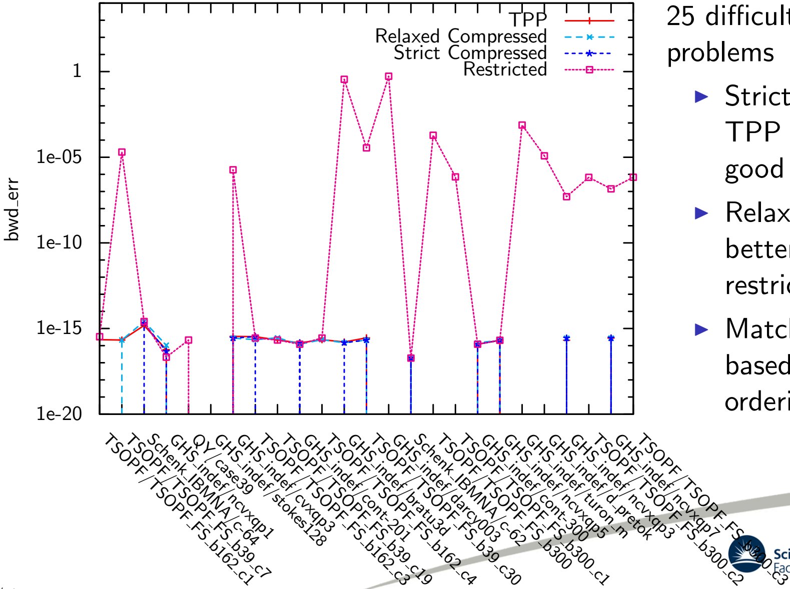
Results: numerical stability



25 difficult problems

- Strict and TPP always good
- Relaxed better than restricted

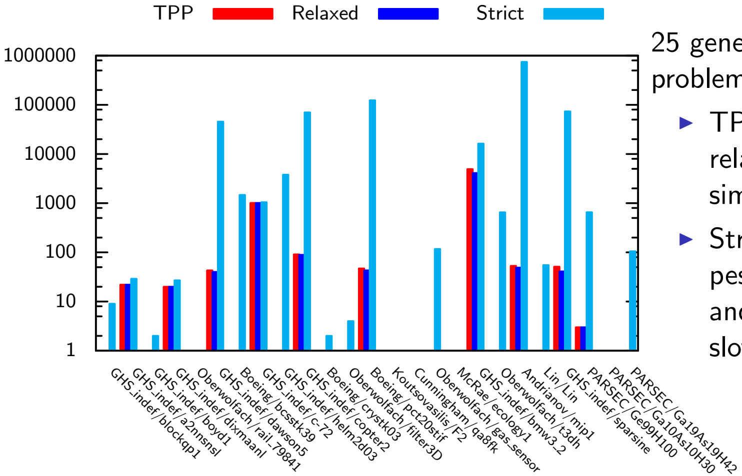
Results: numerical stability



25 difficult problems

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

Results: delays

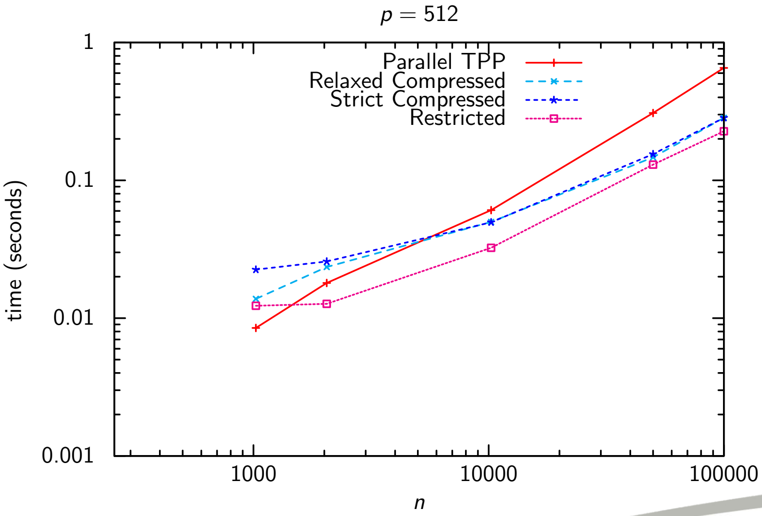


25 general problems

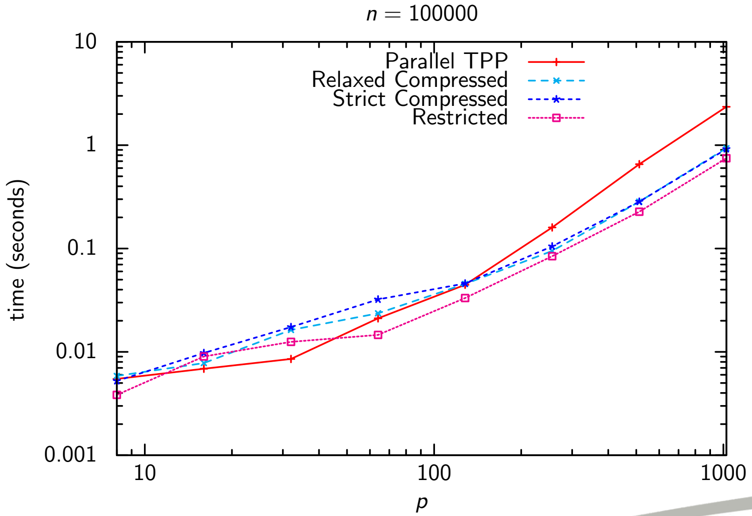
- TPP and relaxed very similar
- Strict very pessimistic and hence slow

Science & Technology Facilities Council

Results: speed $p = 512$



Results: speed $n = 100000$



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

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

New-style solver

- Factorize without pivoting and check L
- If too large, roll-back factorization and...
- ...use compressed pivoting to minimize communication

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

New-style solver

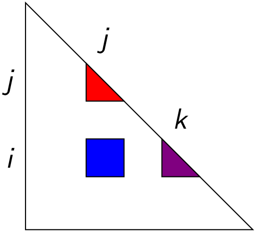
- Factorize without pivoting and check L
- If too large, roll-back factorization and...
- ...use compressed pivoting to minimize communication

Now do it on a GPU!

Science & Technology Facilities Council

Thank you!

Stability



- What if diagonal block is **singular**?
- What if off-diagonal entries **much larger** than diagonal entries?

Then factorization is **not** backwards stable