Scalable pivoting strategies and orderings for sparse symmetric indefinite problems

Stéphane Pralet
CNRS-IRIT, Toulouse

in collaboration with
Iain S. Duff
CERFACS, Toulouse and RAL, UK

CSC05, June 21-23, 2005

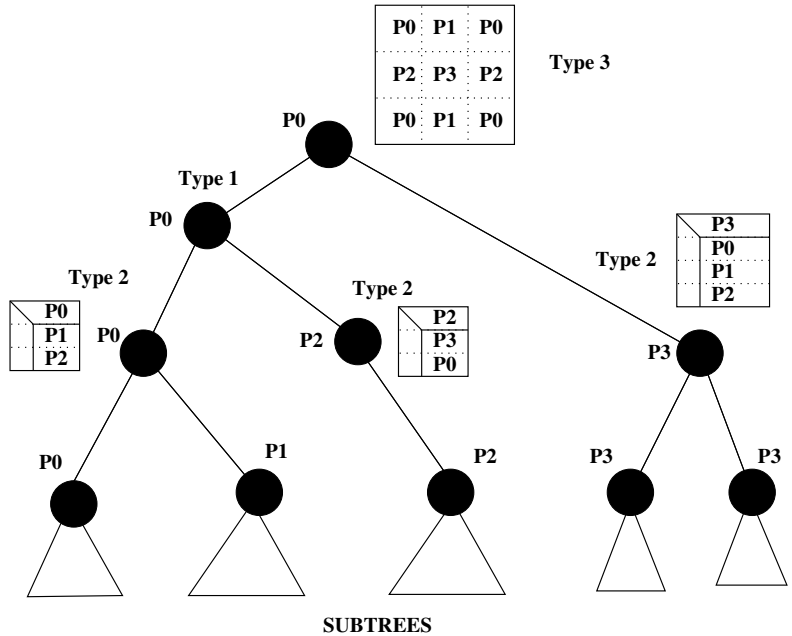
Outline

Sparse LDL^T factorization

- Sparse LDL^T factorization of a sparse symmetric matrix A, where:
  - L, lower triangular
  - D, a block diagonal matrix with 1×1 and 2×2 blocks
- Analysis
  - symmetric ordering to decrease fill-in in the factors
  - symmetric preprocessing techniques to improve the numerical behaviour of the factorization: preselection of "good" 1×1 and 2×2 pivots.
- Factorization
  - numerical pivoting for symmetric indefinite matrices. `MA57` uses Duff-Reid algorithm:
    - stable 1×1 pivots if $|a_{kk}| \geq u \max_j |a_{kj}|$.
    - stable 2×2 pivots if $|P^{-1}| \begin{pmatrix} \max_{k\neq p,q} |a_{pk}| \\ \max_{k\neq p,q} |a_{qk}| \end{pmatrix} \leq \begin{pmatrix} 1/u \\ 1/u \end{pmatrix}$.
  - static pivoting for symmetric indefinite matrices. Existing approaches SuperLU_dist [*Li and Demmel*] (LU factorization), PARDISO [*Schenck and Gärtner*] (modified Bunch-Kaufman)

Multifrontal approach

The assembly tree describes the dependency of computation



A frontal matrix is associated with each node and pivots can be chosen among FSV:

$$\begin{array}{l} \\ FSV \text{ rows} \rightarrow \\ PSV \text{ rows} \rightarrow \end{array} \begin{array}{c} FSV \text{ columns} \quad PSV \text{ columns} \\ \downarrow \quad \downarrow \\ \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \end{array}$$

FSV: fully summed variables / PSV: partially summed variables

Our previous work

- Adaptation of maximum weighted matching techniques `MC64` [*Duff and Koster*]
- Scaling: `MC64` scaling modification, `MC64SYM`
  $\longrightarrow$ significantly improves the factorization time
- Ordering:
  - 1×1 and 2×2 pivots selection strategy suggested by [*Duff and Gilbert*]
  - Compress the graph corresponding to detected 2×2 and 1×1 pivots and weight nodes of compressed graph accordingly
  - Ordering on the compressed graph
    Apply a symmetric ordering to the weighted compressed matrix
    Expand the ordering to A
    $\longrightarrow$ reliability of the forecast pivots
    $\longrightarrow$ the ordering works on a coarse structure which may degrade its quality

Constrained ordering: main ideas

- Consider the 1×1 and 2×2 pivots defined by the symmetric matching
- To offer more choices to the ordering: split some 2×2 pivots into two 1×1 pivots if it can be eliminated as a succession of two 1×1
- But we still want to ensure numerically good pivots: when a pivot is split, optionally add a precedence relationship between the new 1×1 pivots

For a 2×2 pivot $P = \begin{pmatrix} p_1 & offdag \\ offdag & p_2 \end{pmatrix}$, 4 different cases:

Constrained ordering: main ideas

- Consider the 1×1 and 2×2 pivots defined by the symmetric matching
- To offer more choices to the ordering: split some 2×2 pivots into two 1×1 pivots if it can be eliminated as a succession of two 1×1
- But we still want to ensure numerically good pivots: when a pivot is split, optionally add a precedence relationship between the new 1×1 pivots

For a 2×2 pivot $P = \begin{pmatrix} p_1 & offdag \\ offdag & p_2 \end{pmatrix}$, 4 different cases:

1 if p_1 and p_2 small $\Rightarrow$ do not split P

Constrained ordering: main ideas

- Consider the 1×1 and 2×2 pivots defined by the symmetric matching
- To offer more choices to the ordering: split some 2×2 pivots into two 1×1 pivots if it can be eliminated as a succession of two 1×1
- But we still want to ensure numerically good pivots: when a pivot is split, optionally add a precedence relationship between the new 1×1 pivots

For a 2×2 pivot $P = \begin{pmatrix} p_1 & offdag \\ offdag & p_2 \end{pmatrix}$, 4 different cases:

1. if p_1 and p_2 small $\Rightarrow$ do not split P
2. if both p_1 and p_2 large $\Rightarrow$ split P into two 1×1 pivots p_1 and p_2 and no precedence relationship

Constrained ordering: main ideas

- Consider the 1×1 and 2×2 pivots defined by the symmetric matching
- To offer more choices to the ordering: split some 2×2 pivots into two 1×1 pivots if it can be eliminated as a succession of two 1×1
- But we still want to ensure numerically good pivots: when a pivot is split, optionally add a precedence relationship between the new 1×1 pivots

For a 2×2 pivot $P = \begin{pmatrix} p_1 & offdag \\ offdag & p_2 \end{pmatrix}$, 4 different cases:

1. if p_1 and p_2 small $\Rightarrow$ do not split P
2. if both p_1 and p_2 large $\Rightarrow$ split P into two 1×1 pivots p_1 and p_2 and no precedence relationship
3. if p_1 large and p_2 small (the order of elimination is unique) $\Rightarrow$ split P into two 1×1 pivots p_1 and p_2 and force the ordering to eliminate p_1 before p_2: create precedence relationship $p_1\, BEFORE\, p_2$

Constrained ordering: main ideas

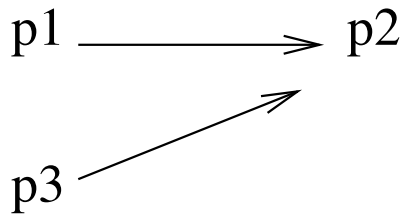
- Consider the 1×1 and 2×2 pivots defined by the symmetric matching
- To offer more choices to the ordering: split some 2×2 pivots into two 1×1 pivots if it can be eliminated as a succession of two 1×1
- But we still want to ensure numerically good pivots: when a pivot is split, optionally add a precedence relationship between the new 1×1 pivots

For a 2×2 pivot $P = \begin{pmatrix} p_1 & offdag \\ offdag & p_2 \end{pmatrix}$, 4 different cases:

1. if p_1 and p_2 small $\Rightarrow$ do not split P
2. if both p_1 and p_2 large $\Rightarrow$ split P into two 1×1 pivots p_1 and p_2 and no precedence relationship
3. if p_1 large and p_2 small (the order of elimination is unique) $\Rightarrow$ split P into two 1×1 pivots p_1 and p_2 and force the ordering to eliminate p_1 before p_2: create precedence relationship $p_1\, BEFORE\, p_2$
4. if p_2 large and p_1 small ...

Relaxed constrained ordering

- Add more possible relationships between variables



permutations allowed:

$p1, p2, p3,$

$p3, p2, p1,$

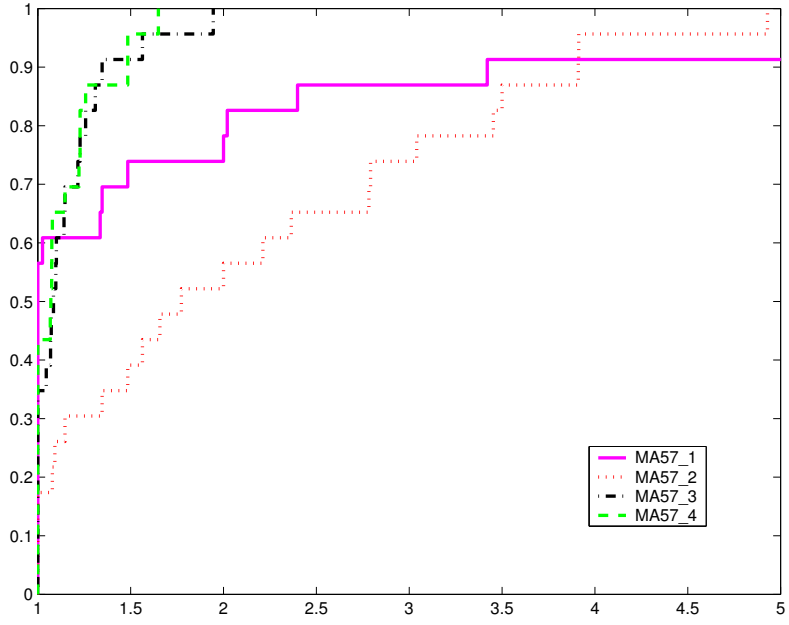
...

Experimental results: augmented systems

`AMF` based ordering / `MC64SYM` scaling used

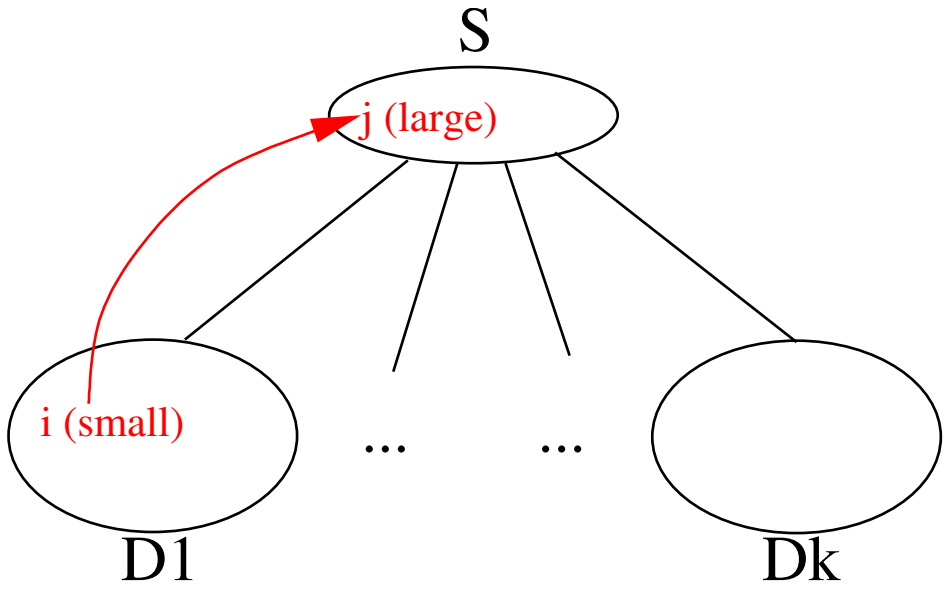
- `MA57_1`: standard factorization
- `MA57_2`: ordering based on the compressed graph
- `MA57_3`: constrained ordering
- `MA57_4`: relaxed constrained ordering

Factorization CPU time



(relaxed) constrained approach is faster

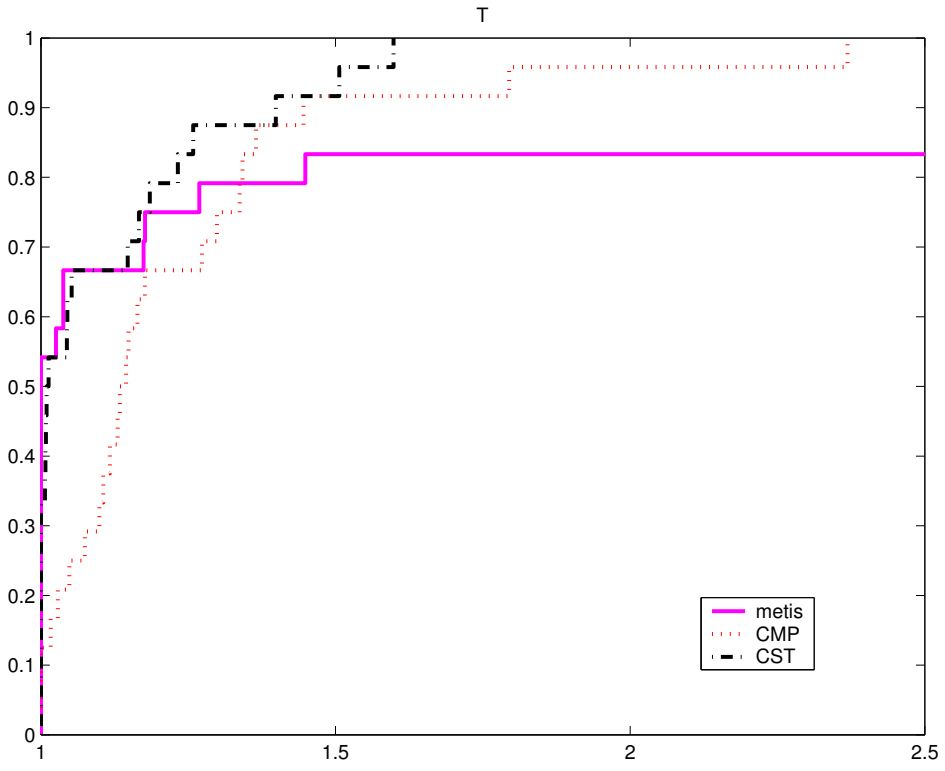
Constrained ordering and partitioner



Compute a partition of the graph
for each domain D_k **do**
  compute $cst_k = \{i \in D_k$ such that $j BEFORE i$ and $j \in S\}$
end for
$S = S \cup_k cst_k$
for each domain D_k **do**
  $D_k = D_k \setminus cst_k$
end for
Apply this algorithm recursively on each domain

Experimental results: factorization time

METIS based ordering / MC64SYM scaling used



- Only slight improvement
- Future work: take the constraints into account when refining the separators

Combining numerical and static pivoting: algorithm

Phase 1: Eliminate as many 1×1 and 2×2 pivots as possible using Duff-Reid algorithm with threshold u.

Combining numerical and static pivoting: algorithm

Phase 1: Eliminate as many 1×1 and 2×2 pivots as possible using Duff-Reid algorithm with threshold u.
Phase 2:
while $FSV \neq \emptyset$ **do**
  Let $i \in FSV$.
  Let $j = arg\max_{k\in FSV\setminus\{i\}} |a_{ik}|$ and P the 2×2 block associated to i and j.

end while

Combining numerical and static pivoting: algorithm

Phase 1: Eliminate as many 1×1 and 2×2 pivots as possible using Duff-Reid algorithm with threshold u.
Phase 2:
while $FSV \neq \emptyset$ **do**
  Let $i \in FSV$.
  Let $j = arg\max_{k\in FSV\setminus\{i\}} |a_{ik}|$ and P the 2×2 block associated to i and j.
  Choose between 1×1 **or** 2×2 **pivoting:**

end while

Combining numerical and static pivoting: algorithm

Phase 1: Eliminate as many 1×1 and 2×2 pivots as possible using Duff-Reid algorithm with threshold u.
Phase 2:
while $FSV \neq \emptyset$ **do**
  Let $i \in FSV$.
  Let $j = arg\max_{k\in FSV\setminus\{i\}} |a_{ik}|$ and P the 2×2 block associated to i and j.
  Choose between 1×1 **or** 2×2 **pivoting:**
  if $\min\{g_1(i), g_2(i,j)\} < 1/\sqrt{\epsilon}$ **then** /*Case 1: growth factor comparison*/
    if $g_2(i,j) < g_1(i)$ **then**
      Perform elimination using (i,j) as a 2×2 pivot.
    else
      Perform elimination using i as a 1×1 pivot.
    end if
    continue
  end if

end while

Combining numerical and static pivoting: algorithm

Phase 1: Eliminate as many 1×1 and 2×2 pivots as possible using Duff-Reid algorithm with threshold u.
Phase 2:
while $FSV \neq \emptyset$ **do**
  Let $i \in FSV$.
  Let $j = arg\max_{k\in FSV\setminus\{i\}} |a_{ik}|$ and P the 2×2 block associated to i and j.
  Choose between 1×1 **or** 2×2 **pivoting:**
  if $\min\{g_1(i), g_2(i,j)\} < 1/\sqrt{\epsilon}$ **then** /*Case 1: growth factor comparison*/
    if $g_2(i,j) < g_1(i)$ **then**
      Perform elimination using (i,j) as a 2×2 pivot.
    else
      Perform elimination using i as a 1×1 pivot.
    end if
    continue
  end if
  if $||A||_M \ \min\{1/|a_{ii}|, ||P^{-1}||_\infty\} < 1/\sqrt{\epsilon}$ **then** /*Case 2: pivot size comparison*/
    if $1/|a_{ii}| > ||P^{-1}||_\infty$ **then**
      Perform elimination using (i,j) as a 2×2 pivot.
    else
      Perform elimination using i as a 1×1 pivot.
    end if
    continue
  end if

end while

Combining numerical and static pivoting: algorithm

Phase 1: Eliminate as many 1×1 and 2×2 pivots as possible using Duff-Reid algorithm with threshold u.
Phase 2:
while $FSV \neq \emptyset$ **do**
  Let $i \in FSV$.
  Let $j = arg\max_{k\in FSV\setminus\{i\}} |a_{ik}|$ and P the 2×2 block associated to i and j.
  Choose between 1×1 **or** 2×2 **pivoting:**
  if $\min\{g_1(i), g_2(i,j)\} < 1/\sqrt{\epsilon}$ **then** /***Case 1: growth factor comparison***/
    if $g_2(i,j) < g_1(i)$ **then**
      Perform elimination using (i,j) as a 2×2 pivot.
    else
      Perform elimination using i as a 1×1 pivot.
    end if
    continue
  end if
  if $||A||_M \ \min\{1/|a_{ii}|, ||P^{-1}||_\infty\} < 1/\sqrt{\epsilon}$ **then** /***Case 2: pivot size comparison***/
    if $1/|a_{ii}| > ||P^{-1}||_\infty$ **then**
      Perform elimination using (i,j) as a 2×2 pivot.
    else
      Perform elimination using i as a 1×1 pivot.
    end if
    continue
  end if
  $a_{ii} = s(a_{ii})\sqrt{\epsilon}\,||A||_M$ /***Case 3: static pivoting***/
  Perform elimination using i as a 1×1 pivot.
end while

Component-wise backward error and number of tiny pivots

Matrix	numerical pivoting		`mixedSEQ` pivoting strategy			
	it. 0	it. 1	it. 0	it. 1	it. 2	tiny
BRAINPC2	1.6e-15		2.1e-08			12932
BRATU3D	2.0e-09		9.2e-06			8429
CONT-201	8.8e-11		1.0e-05			27470
CONT-300	7.6e-11		2.1e-05			67864
cvxqp3	5.2e-11		8.5e-06			6277
DTOC	2.1e-16		8.3e-07			9790
mario001	6.3e-15		3.1e-08			10305
NCVXQP1	4.6e-14		4.9e-13			3619
NCVXQP5	2.0e-11		2.0e-08			8402
NCVXQP7	9.6e-10		4.9e-06			31043
SIT100	4.4e-15		2.0e-08			1388
stokes128	1.1e-14		4.2e-14			12738
stokes64	4.3e-15		1.6e-13			3106

Component-wise backward error and number of tiny pivots

Matrix	numerical pivoting		`mixedSEQ` pivoting strategy			
	it. 0	it. 1	it. 0	it. 1	it. 2	tiny
BRAINPC2		1.0e-15		5.7e-15		12932
BRATU3D		1.7e-16		2.2e-11		8429
CONT-201		1.6e-16		9.4e-09		27470
CONT-300		1.9e-16		2.7e-09		67864
cvxqp3		2.7e-16		1.2e-12		6277
DTOC		2.7e-20		2.1e-13		9790
mario001		1.3e-16		2.5e-13		10305
NCVXQP1		1.7e-17		3.2e-15		3619
NCVXQP5		2.0e-16		6.7e-11		8402
NCVXQP7		2.2e-16		1.4e-12		31043
SIT100		1.4e-16		5.8e-15		1388
stokes128		5.5e-16		2.0e-15		12738
stokes64		1.5e-15		2.3e-14		3106

Component-wise backward error and number of tiny pivots

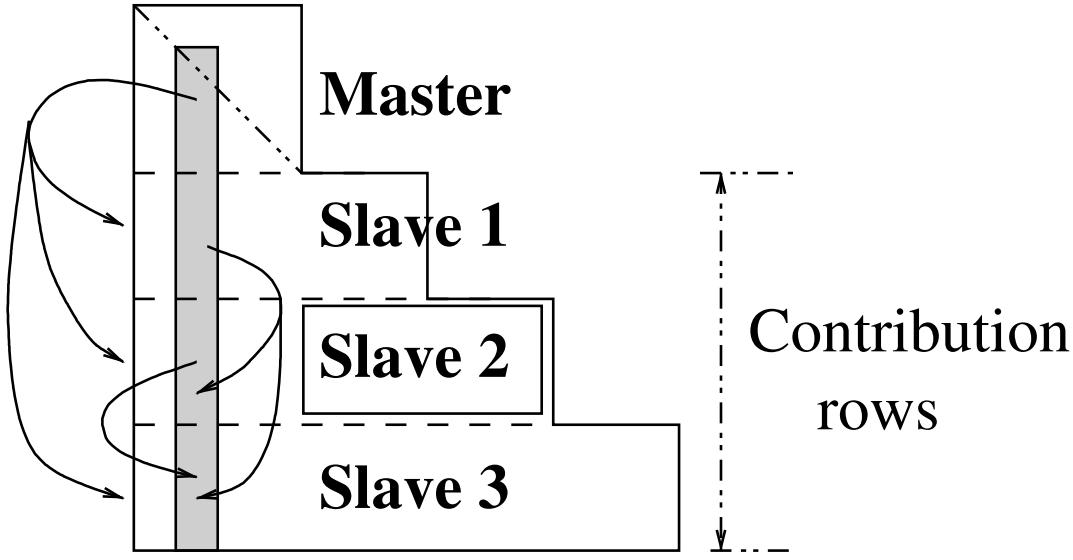
Matrix	numerical pivoting		`mixedSEQ` pivoting strategy			
	it. 0	it. 1	it. 0	it. 1	it. 2	tiny
BRAINPC2		1.0e-15		5.7e-15	9.8e-16	12932
BRATU3D		1.7e-16		2.2e-11	1.7e-16	8429
CONT-201		1.6e-16		9.4e-09	4.9e-09	27470
CONT-300		1.9e-16		2.7e-09	2.5e-09	67864
cvxqp3		2.7e-16		1.2e-12	3.4e-16	6277
DTOC		2.7e-20		2.1e-13	1.9e-15	9790
mario001		1.3e-16		2.5e-13	1.3e-16	10305
NCVXQP1		1.7e-17		3.2e-15	2.6e-17	3619
NCVXQP5		2.0e-16		6.7e-11	2.7e-14	8402
NCVXQP7		2.2e-16		1.4e-12	2.2e-16	31043
SIT100		1.4e-16		5.8e-15	1.5e-16	1388
stokes128		5.5e-16		2.0e-15	1.7e-15	12738
stokes64		1.5e-15		2.3e-14	2.2e-14	3106

Factorization time (in seconds), number of delayed pivots, size of the factors

Matrix	Nmb delayed	Factorization time		Size of the factors	
		numSEQ	mixedSEQ	numSEQ	mixedSEQ
BRAINPC2	14267	0.18	0.11	656	322
BRATU3D	90052	34.2	9.24	11484	5569
CONT-201	71296	5.51	1.94	8820	4304
CONT-300	183306	21.1	6.08	23838	10714
cvxqp3	30519	9.73	3.08	4740	2301
DTOC	29478	29.1	0.41	4714	187
mario001	15463	0.28	0.23	817	575
NCVXQP1	12463	2.69	1.29	2235	1327
NCVXQP5	16703	25.7	23.0	13365	11205
NCVXQP7	195973	195.	71.6	37683	19367
SIT100	2710	0.13	0.11	483	417
stokes128	18056	1.14	1.06	3437	2753
stokes64	4292	0.33	0.29	736	577

`numSEQ`: numerical pivoting strategy. `mixedSEQ`: combination of static and numerical pivoting.

Type 2 nodes



An example of parallelization of a type 2 node (multifrontal context)

Efficient parallel approach:

- The master does not compute the off diagonal part $\longrightarrow$ shorter critical path
- No synchronization between a master and its slaves

Limiting the areas of pivot checking

PSV: partially summed variables
FSV: fully summed variables

- growth factor of a 1×1 pivot:

$$g_1(i) = \frac{\max_{k \in (FSV \cup PSV) \setminus \{i\}} |a_{ik}|}{|a_{ii}|}$$

Limiting the areas of pivot checking

PSV: partially summed variables
FSV: fully summed variables

- growth factor of a 1×1 pivot:

$$g_1(i) = \frac{\max_{k \in (FSV \qquad) \setminus \{i\}} |a_{ik}|}{|a_{ii}|}$$

Limiting the areas of pivot checking

PSV: partially summed variables
FSV: fully summed variables

- growth factor of a 1×1 pivot:

$$g_1(i) = \frac{\max\{\max_{k \in (FSV \qquad) \setminus \{i\}} |a_{ik}|, \sqrt{\epsilon}\, ||A||_M\}}{|a_{ii}|}$$

Limiting the areas of pivot checking

PSV: partially summed variables
FSV: fully summed variables

- growth factor of a 1×1 pivot:

$$g_1(i) = \frac{\max\{\max_{k\in(FSV \qquad)\setminus\{i\}} |a_{ik}|, \sqrt{\epsilon}\,||A||_M\}}{|a_{ii}|}$$

- growth factor of a 2×2 pivot:

$$g_2(i,j) = \left|\left| |P^{-1}| \begin{pmatrix} \max\{\max_{k\in FSV\setminus\{i,j\}} |a_{ik}|, \sqrt{\epsilon}\,||A||_M\} \\ \max\{\max_{k\in FSV\setminus\{i,j\}} |a_{jk}|, \sqrt{\epsilon}\,||A||_M\} \end{pmatrix} \right|\right|_\infty .$$

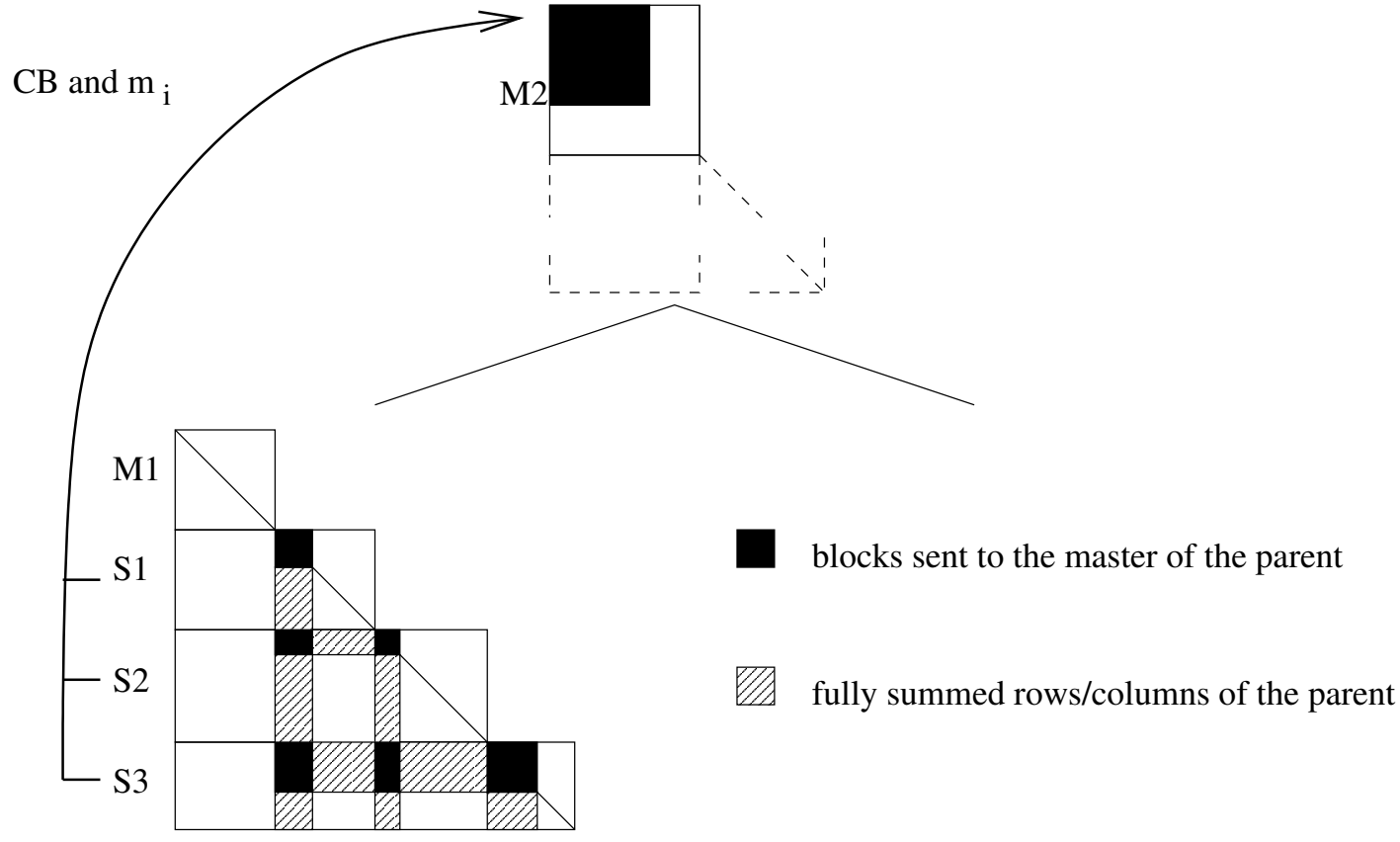
Limiting the areas of pivot checking

Matrix	Iteration 0		Iteration 1	
	numSEQ	numBPAR	numSEQ	numBPAR
BRAINPC2	1.6e-15	1.6e-15	1.0e-15	1.0e-15
BRATU3D	2.0e-09	4.8e-01	1.7e-16	4.6e-01
CONT-201	8.8e-11	1.3e-01	1.6e-16	1.2e-01
CONT-300	7.6e-11	1.3e-01	1.9e-16	7.7e-02
cvxqp3	5.2e-11	4.8e-06	2.7e-16	2.5e-06
DTOC	2.1e-16	2.1e-16	2.7e-20	2.7e-20
mario001	6.3e-15	6.3e-15	1.3e-16	1.3e-16
NCVXQP1	4.6e-14	9.2e-14	1.7e-17	3.0e-17
NCVXQP5	2.0e-11	2.5e-10	2.0e-16	2.0e-16
NCVXQP7	9.6e-10	5.9e-08	2.2e-16	1.2e-07
SIT100	4.4e-15	4.4e-15	1.4e-16	1.4e-16
stokes128	1.1e-14	1.1e-14	5.5e-16	5.5e-16
stokes64	4.3e-15	4.3e-15	1.5e-15	1.5e-15

Component-wise backward error of strategies with numerical pivoting.
numSEQ: sequential approach. numBPAR: basic parallel approach.

Cheap estimation of non fully summed

Use information coming from the slaves of the children:



Each slave, s, of child, c, computes the largest entry in each fully summed row, i,of parent, $m_i(s)$, and sends this to master of parent

Cheap estimation of non fully summed

Estimation of the max off-diagonal entry in row i:

$$M_i = \max\{ \max_{c \text{ child of } p} \quad \max_{s \text{ slave of } c} m_i(s) \quad , \max_k |a_{ik}^{(0)}| \}$$

For a 1×1 pivot:

$$g_1(i) = \frac{\max\{\max_{k \in FSV_p \setminus \{i\}} |a_{ik}|, M_i, \sqrt{\epsilon}\, ||A||_M\}}{|a_{ii}|}.$$

For a 2×2 pivot:

$$g_2(i,j) = \left|\left| |P^{-1}| \begin{pmatrix} \max\{\max_{k \in FSV_p \setminus \{i,j\}} |a_{ik}|, M_i, \sqrt{\epsilon}\, ||A||_M\} \\ \max\{\max_{k \in FSV_p \setminus \{i,j\}} |a_{jk}|, M_j, \sqrt{\epsilon}\, ||A||_M\} \end{pmatrix} \right|\right|_\infty .$$

Cheap estimation of non fully summed

Matrix	Iteration 0		Iteration 1	
	numSEQ	numEPAR	numSEQ	numEPAR
BRAINPC2	1.6e-15	1.6e-15	1.0e-15	1.0e-15
BRATU3D	2.0e-09	7.8e-09	1.7e-16	1.7e-16
CONT-201	8.8e-11	3.0e-10	1.6e-16	2.3e-16
CONT-300	7.6e-11	1.4e-10	1.9e-16	1.7e-16
cvxqp3	5.2e-11	2.2e-10	2.7e-16	2.7e-16
DTOC	2.1e-16	2.1e-16	2.7e-20	2.7e-20
mario001	6.3e-15	6.3e-15	1.3e-16	1.3e-16
NCVXQP1	4.6e-14	5.2e-14	1.7e-17	2.7e-17
NCVXQP5	2.0e-11	5.8e-11	2.0e-16	2.0e-16
NCVXQP7	9.6e-10	1.3e-09	2.2e-16	2.2e-16
SIT100	4.4e-15	4.4e-15	1.4e-16	1.4e-16
stokes128	1.1e-14	1.1e-14	5.5e-16	5.5e-16
stokes64	4.3e-15	4.3e-15	1.5e-15	1.5e-15

Component-wise backward error of strategies with numerical pivoting. `numSEQ`: sequential approach. `numEPAR`: parallel approach using estimations.

Parallel approaches and static pivoting

Use cheap estimation for $g_1(i)$ and $g_2(i,j)$ in the combination of static and numerical pivoting:

Phase 1: Duff-Reid algorithm with threshold u + estimations.
Phase 2:
while $FSV \neq \emptyset$ **do**
  ...
  Choose between 1×1 **or** 2×2 **pivoting:**
  if $\min\{\text{estimations} \rightarrow g_1(i), g_2(i,j)\} < 1/\sqrt{\epsilon}$ **then** /***Case 1***/
    ...
  end if
  ...
end while

Parallel approaches and static pivoting

Matrix	Iteration 0		Iteration 1	
	`mixedPAR`	`mixedPAR_P`	`mixedPAR`	`mixedPAR_P`
BRAINPC2	2.1e-08		5.7e-15	
BRATU3D	1.7e-05		1.3e-10	
CONT-201	1.8e-05		9.4e-09	
CONT-300	1.9e-05		2.6e-09	
cvxqp3	8.0e-06		9.3e-13	
DTOC	8.3e-07		2.1e-13	
mario001	3.1e-08		2.5e-13	
NCVXQP1	3.3e-13		4.4e-15	
NCVXQP5	7.5e-08		1.6e-11	
NCVXQP7	4.3e-06		2.0e-12	
SIT100	2.0e-08		5.8e-15	
stokes128	4.2e-14		2.0e-15	
stokes64	1.6e-13		2.3e-14	

Component-wise backward error of strategies with static pivoting. `mixedPAR`: static pivoting strategy without preprocessing. `mixedPAR_P`: static pivoting strategy with preprocessing.

Parallel approaches and static pivoting

Matrix	Iteration 0		Iteration 1	
	`mixedPAR`	`mixedPAR_P`	`mixedPAR`	`mixedPAR_P`
BRAINPC2	2.1e-08	4.3e-13	5.7e-15	9.5e-16
BRATU3D	1.7e-05	6.9e-08	1.3e-10	1.6e-15
CONT-201	1.8e-05	1.2e-13	9.4e-09	1.7e-16
CONT-300	1.9e-05	2.4e-13	2.6e-09	1.7e-16
cvxqp3	8.0e-06	5.3e-06	9.3e-13	3.2e-14
DTOC	8.3e-07	6.1e-16	2.1e-13	2.6e-20
mario001	3.1e-08	5.9e-08	2.5e-13	3.1e-15
NCVXQP1	3.3e-13	1.6e-14	4.4e-15	6.7e-15
NCVXQP5	7.5e-08	6.8e-12	1.6e-11	2.0e-16
NCVXQP7	4.3e-06	1.6e-08	2.0e-12	2.0e-14
SIT100	2.0e-08	1.2e-07	5.8e-15	1.3e-14
stokes128	4.2e-14	5.9e-14	2.0e-15	4.9e-14
stokes64	1.6e-13	1.6e-13	2.3e-14	2.2e-13

Component-wise backward error of strategies with static pivoting. `mixedPAR`: static pivoting strategy without preprocessing. `mixedPAR_P`: static pivoting strategy with preprocessing.

Concluding remarks, current and future work

- AMF + constraints ordering improves factorization time and memory estimation
- METIS + constraints only small improvements
- Static pivoting is fast but sometimes dangerous
- Numerical preprocessings improve the robustness of static pivoting
- Robust parallel approaches based on cheap estimations $\longrightarrow$ MUMPS
- Use of more sophisticated iterative methods (GMRES, MINRES, ...)