Efficiency Improvement of Panel Codes

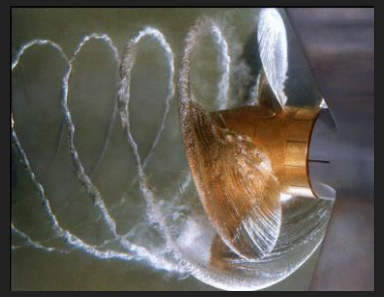
Master Thesis Presentation
10th July 2015
Ang Yun Mei Elisa (4420888)

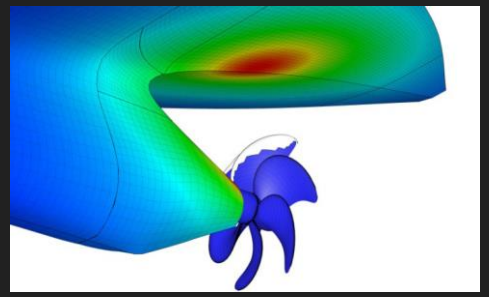
Supervisor:	Dr. ir. M.B. van Gijzen	TU Delft
MARIN supervisor:	Dr. ir. A. van der Ploeg	MARIN
Thesis Committee:	Prof. dr. ir. C. Vuik	TU Delft
	Dr. ir. H.X. Lin	TU Delft

TU Delft

Problem Statement

- MARIN uses Panel Codes to compute flows
- Panel Codes produces a dense linear system of Equation Ax=b
- There's a need to improve the performance of the dense linear solver





TUDelft

Presentation Overview

- Background and current status
- **Strategy 1:** Changing the solver
- **Strategy 3:** Using the hierarchical method to speedup matrix-vector multiplication
- **Strategy 4:** Changing the preconditioner to hierarchical-LU preconditioner
- Conclusion and future work

Background

Before 2012:
Direct Solvers or GMRES with ILU

2012 till now:
GMRES with block Jacobi (Work of M. de Jong)

Sept 2014:
Project Literature Review commence

3 strategies were identified

1. Replacing Solver: GMRES with IDR(s)
2. Updating of current block Jacobi preconditioner to take variable size blocks
3. Using hierarchical method to speed up matrix-vector multiplication

During the course of the project, the forth strategy was found:

4. Hierarchical- LU Preconditioner

TU Delft

Test matrices

- The same test matrices as what Martijn are used here too:

Name	Size	Real/Complex
Steadycav1	4620	Real
Steadycav2	4620	Real
Steadycav3	4620	Real
Steadycav4	4649	Real
Passcal	4400	Real
FATIMA_7894	7894	Complex
FATIMA_20493	20493	Complex

TU Delft

Current Status

- Code from work of Martijn de Jong were ran in our system to produce the following baseline results (the block Jacobi size resulting in the lowest time was chosen)

Test Matrix	NRHS	Jacobi Block Size	Time in parallel (4 cores, openmp)	Time in Serial
FATIMA_20493	1	4000	87.62 s	239.8 s
	7	4000	211.49 s	Not ran
FATIMA_7894	1	1000	6.36 s	21.3 s
	7	1000	25.74 s	Not ran
PASSCAL	1	500	0.72 s	2.2 s
Steadycav1	1	500	0.57 s	1.7 s
Steadycav2	1	500	0.60 s	1.8 s
Steadycav3	1	500	0.67 s	1.9 s
Steadycav4	1	500	0.67 s	2.0 s

Strategy 1: Replacing GMRES with IDR(s)

- Brief overview of GMRES & IDR(s)
- Results

TU Delft

GMRES

- By Yousef Saad and Martin H. Schultz in 1986
- Advantages
  - Optimality
  - 1 matrix vector multiplication required per iteration
- Disadvantages
  - Long recurrence
  - For practical reason, GMRES with restart is often implemented

Extracted from: http://www-users.cs.umn.edu/~saad/ & http://cpsc.yale.edu/people/martin-schultz

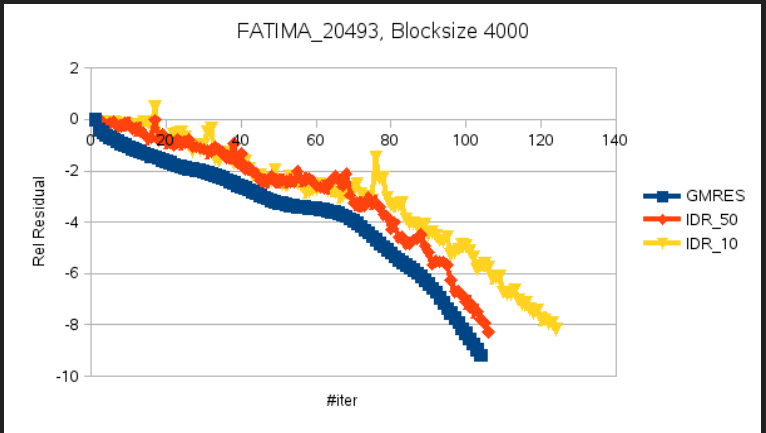
IDR(s)

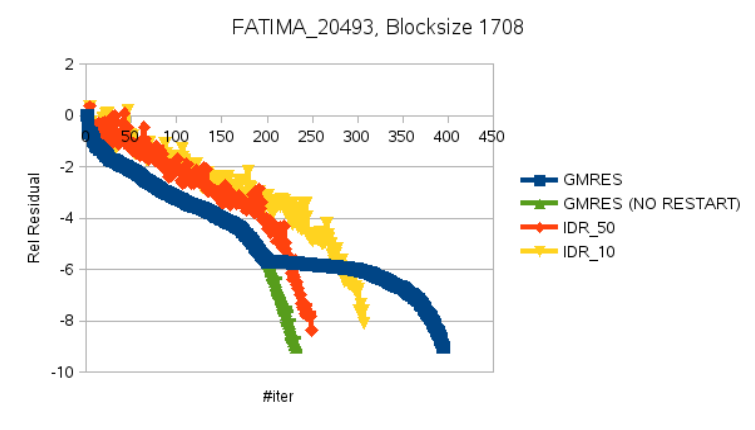
- By Peter Sonneveld and Martin van Gijzen in 2008.
- Advantages
  - Short recurrence
  - 1 matrix vector multiplication required per iteration
- Disadvantages
  - Non optimal
  - Hence, expected to require more iterations for convergence

Extracted from:
http://ta.twi.tudelft.nl/nw/users/gijzen/IDR.html

RESULTS of GMRES vs IDR(s)

Matrix	Blocksize	GMRES				IDR(s)			
		Wall clock time (s)		#iter	Rel error	Wall clock time (s)		#iter	Rel error
		Solve	Total			Solve	Total		
FATIMA_20493	1708	116.08	122.66	393	1.49E-06	73.94	80.45	260	6.20E-07
	4000	34.44	87.62	103	1.17E-07	36.57	89.75	110	2.45E-07
	6000	20.64	287.64	60	2.27E-07	23.07	292.51	66	1.36E-07





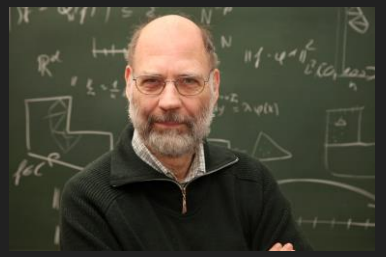
TUDelft

Strategy 3: Hierarchical method to speed up matrix vector multiplication

- Theory behind hierarchical matrices
- Low rank approximation
- How matrix-vector multiplication is speed up
- Results

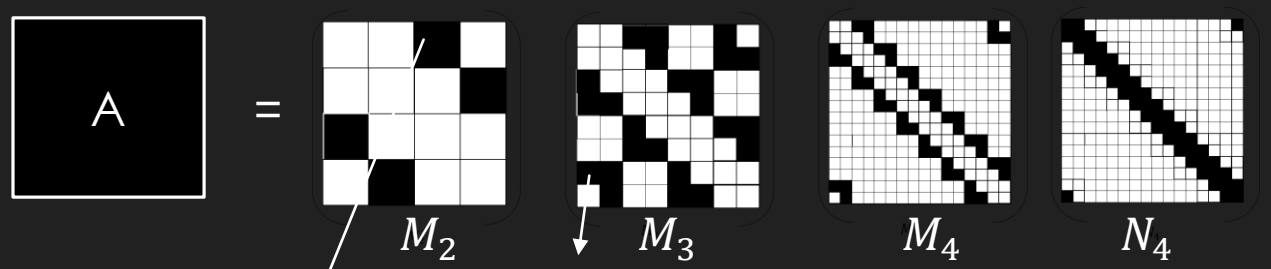
TU Delft

What is the hierarchical form of a matrix?



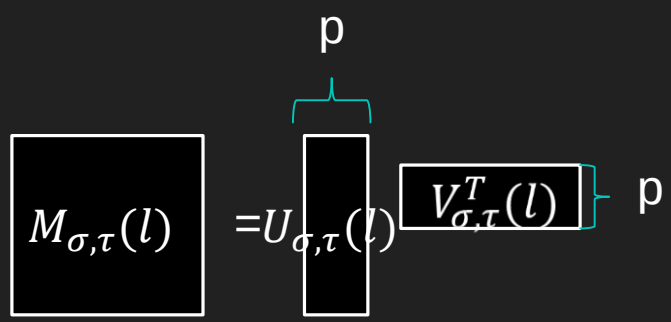
Extracted from:
www.mis.mpg.de

- Hierarchical matrices introduced by Wolfgang Hackbush in 1999
- Idea:
  - Matrices from BEM has a hierarchical structure



$M_{1,3}(2)$

$M_{7,1}(3)$



$$M_{\sigma,\tau}(l) \approx \widetilde{M}_{\sigma,\tau}(l) = \sum_{k=1}^{p} u_k v_k^T = U_{\sigma,\tau} V_{\sigma,\tau}^T$$

Reduces total number of elements in a block from N^2 to $2Np$

- Reduces complexity of matrix-vector multiplication from $O(N^2)$ to $O(NlogN)$

Low rank approximation

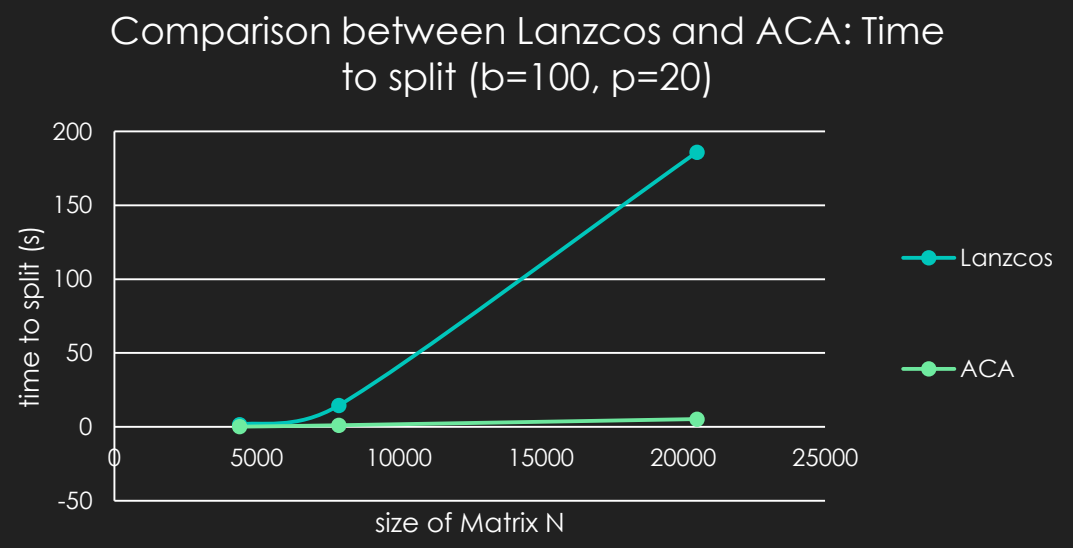
- Methods to perform low rank approximation
  - Singular Value Decomposition (SVD) $O(N^3)$

$$M = [U]\begin{bmatrix}\sigma_1 & & \\ & \ddots & \\ & & \end{bmatrix}[V]^T$$

  - Lanzcos Bidiagonalization $O(N^2)$

$$M = [U]\begin{bmatrix}b_{11} & & \\ b_{21} & \ddots & \\ & \ddots & \end{bmatrix}[V]^T$$

  - Adaptive Cross Appoximation (ACA) $O(N)$



Adaptive Cross Approximation (ACA)

- Introduced by Mario Bebendorf in 2000
- Approximate low rank blocks with outer products
- Main equations

$$\widetilde{M}_{\sigma,\tau}(l) = \sum_{k=1}^{p} u_k v_k^T$$

$$M = R + S,\ R_0 = M,\ S_0 = 0$$

$$\gamma_{k+1} = \frac{1}{R_k(i_{k+1}, j_{k+1})}$$

$$R_{k+1} = R_k - \gamma_{k+1} R_k\big(:, j_{k+1)}\big) R_k(i_{k+1}, :)$$

$$S_{k+1} = S_k + \gamma_{k+1} R_k\big(:, j_{k+1)}\big) R_k(i_{k+1}, :)$$

TUDelft

Adaptive Cross Approximation (ACA)

- Looking at the main equations in further details

$$M = R + S$$

$$\gamma_{k+1} = \frac{1}{R_k(i_{k+1}, j_{k+1})}$$

$$R_{k+1} = R_k - \boxed{\gamma_{k+1} R_k(:, j_{k+1)}) R_k(i_{k+1}, :)}$$

$$S_{k+1} = S_k + \gamma_{k+1} R_k(:, j_{k+1)}) R_k(i_{k+1}, :)$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

R_0

$R_k(:, j_{k+1}) = R_0(:, 2)$

$\gamma_{k+1} = \frac{1}{4}$

$R_k(i_{k+1}, :) = R_0(2, :)$

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

S_0

$$\gamma_{k+1} R_k(:, j_{k+1)}) R_k(i_{k+1}, :) = \frac{1}{4}\begin{bmatrix} 2 \\ 4 \end{bmatrix} \begin{bmatrix} 3 & 4 \end{bmatrix} = \begin{bmatrix} \frac{3}{2} & 2 \\ 3 & 4 \end{bmatrix}$$

$$\begin{bmatrix} -\frac{1}{2} & 0 \\ 0 & 0 \end{bmatrix}$$

R_1

$$\begin{bmatrix} \frac{3}{2} & 2 \\ 3 & 4 \end{bmatrix}$$

S_1

Adaptive Cross Approximation (ACA)

- Obtaining the low rank approximation $\mathrm{M} \approx \widetilde{M} = \sum_{k=1}^{p} u_k v_k^T = S_p$

$$M = R + S,\ R_0 = M,\ S_0 = 0$$

$$\gamma_{k+1} = \frac{1}{R_k(i_{k+1}, j_{k+1})}$$

$$R_{k+1} = R_k - \gamma_{k+1} R_k\big(:, j_{k+1)}\big) R_k(i_{k+1}, :)$$

$$S_{k+1} = S_k + \gamma_{k+1} R_k\big(:, j_{k+1)}\big) R_k(i_{k+1}, :)$$

$$u_{k+1} = R_k(:, j_{k+1})$$
$$v_{k+1} = \gamma_{k+1} R_k(i_{k+1}, :)$$

$$M = R + S,\ R_0 = M,\ S_0 = 0$$

$$\gamma_{k+1} = \frac{1}{R_k(i_{k+1}, j_{k+1})}$$

$$R_{k+1} = R_k - u_{k+1} v_{k+1}$$

$$\boxed{S_{k+1} = S_k + u_{k+1} v_{k+1}}$$

TUDelft

Adaptive Cross Approximation (ACA)

- Choice of pivot rows and columns
  - Lowest residual if most dominant element of R_k is always chosen -> complete pivoting
    - Expensive $O(N^2)$ operation
  - Instead, use partial pivoting
    - Randomly set i_1
    - Choose the largest element from that row -> j_1
    - Choose the largest element from that col -> i_2
    - Whenever we have all zeros, choose the next available row/col

$$\begin{bmatrix} 1 & 2 \\ 3 & \boxed{4} \end{bmatrix} \quad i_1 = 2, j_1 = 2$$

$$\underset{R_0}{\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}} \quad i_1 = 1, \; j_1 = 2 \longrightarrow \underset{R_1}{\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}} \quad j_2 = 1, \; i_2 = 2 \longrightarrow \underset{R_2}{\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}}$$

Adaptive Cross Approximation (ACA)

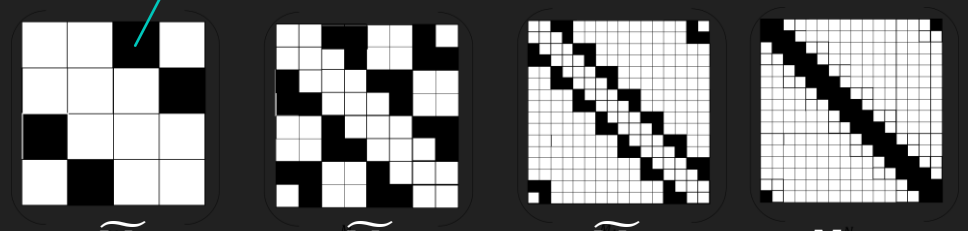
- Summary of the ACA algorithm
  - Initialization: $R_0 = M, S_0 = 0, i_1 = 1$
  - For k=0,1,2,…, p
    - Find j_{k+1}
    - $\gamma_{k+1} = R_k(i_{k+1}, j_{k+1})$
    - $v_{k+1} = \gamma_{k+1} R_k(i_{k+1}, :)$
    - $u_{k+1} = R_k(:, j_{k+1})$
    - Compute new $R_{k+1} = R_k - u_{k+1} v_{k+1}$
  - If $\|R_p\| \leq \varepsilon \|M\|$, then M is low rank. Its rank p approximation is $\sum_{k=1}^{p} u_k v_k^T$

TUDelft

Hierarchical form of the matrix

- With ACA, a matrix A can be brought into its hierarchical form

$U_{1,3}(2)V_{1,3}^T(2)$



$A \approx \widetilde{M}_2 + \widetilde{M}_3 + \widetilde{M}_4 + N_4$

Hierarchical matrix-vector multiplication

- Matrix Vector multiplication can be approximated:

$$Ax \approx \sum_{l=2}^{levels} \boxed{\widetilde{M}_l x} + \boxed{N_{levels} x}$$

$$\widetilde{M}_l x = \begin{bmatrix} (\widetilde{M}_l x)_1 \\ \vdots \\ (\widetilde{M}_l x)_\sigma \\ \vdots \\ (\widetilde{M}_l x)_{2^l} \end{bmatrix} = \begin{bmatrix} \sum_{\tau=1}^{2^l} U_{1,\tau} V_{1,\tau}^T x_\tau \\ \vdots \\ \sum_{\tau=1}^{2^l} U_{2^l,\tau} V_{2^l,\tau}^T x_\tau \end{bmatrix}$$

At most $O(N)$ non zero elements
$\rightarrow O(N)$

Each $U_{\sigma,\tau}(l)$ and $V_{\sigma,\tau}(l)$ has $\frac{N}{2^l} \times p$ non zero elements

Each level, $O(2^l)$ admissible blocks

There are at most $logN$ levels

$\rightarrow O(NlogN)$

$O(NlogN)$

Results

FATIMA_20493								
Block Jacobi block size	Wall clock time (s)		#iter	Rel error	Wall clock time (s)		#iter	Rel error
	Solve	Total			Solve	Total		
	IDR(50) with dense matvec				IDR(50) with hierarchical matvec			
1708	248.87	255.46	259	2.47E-07	109.23	123.10	258	6.15E-01
4000	111.30	164.81	109	1.66E-07	53.60	113.88	113	6.15E-01
6000	68.48	335.77	66	1.36E-07	31.68	304.75	65	6.15E-01

Solve time drops by half

But accuracy is unacceptable…

Strategy 4: Hierarchical LU-preconditioner

Theory

Integration with solver

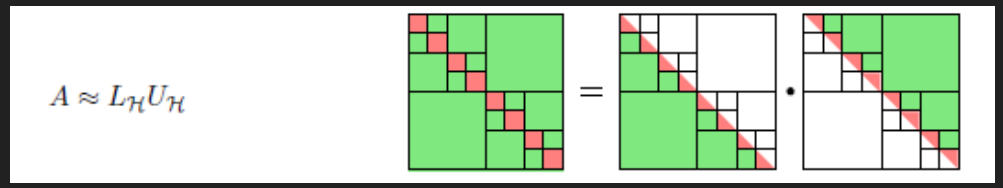
Results in Serial

Results in parallel

TU Delft

Theory

- Idea: decompose a hierarchical matrix A into hierarchical lower and upper triangular matrix L and U



Extracted from: Börm, S., Grasedyck, L. & Hackbush, W. 2005. Hierarchical Matrices (Lecture Notes).

- This is done recursively. Imagine the matrix A split into 4 blocks

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \times \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix} = \begin{bmatrix} L_{11}U_{11} & L_{11}U_{12} \\ L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22} \end{bmatrix}$$

Theory

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \times \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix} = \begin{bmatrix} L_{11}U_{11} & L_{11}U_{12} \\ L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22} \end{bmatrix}$$

- The problem of solving for L and U is divided into four sub problems
  1. $A_{11} = L_{11}U_{11}$ → Recursively do hierarchical LU decomposition
  2. $A_{12} = L_{11}U_{12}$ → Hierarchical Lower Triangular Solver
  3. $A_{21} = L_{21}U_{11}$ → Hierarchical Upper Triangular Solver
  4. $A_{22} - L_{21}U_{12} = L_{22}U_{22}$ → Rounded Subtraction
Recursively do hierarchical LU decomposition

Theory

1. $A_{11} = L_{11}U_{11}$
2. $A_{12} = L_{11}U_{12}$
3. $A_{21} = L_{21}U_{11}$
4. $A_{22} - L_{21}U_{12} = L_{22}U_{22}$

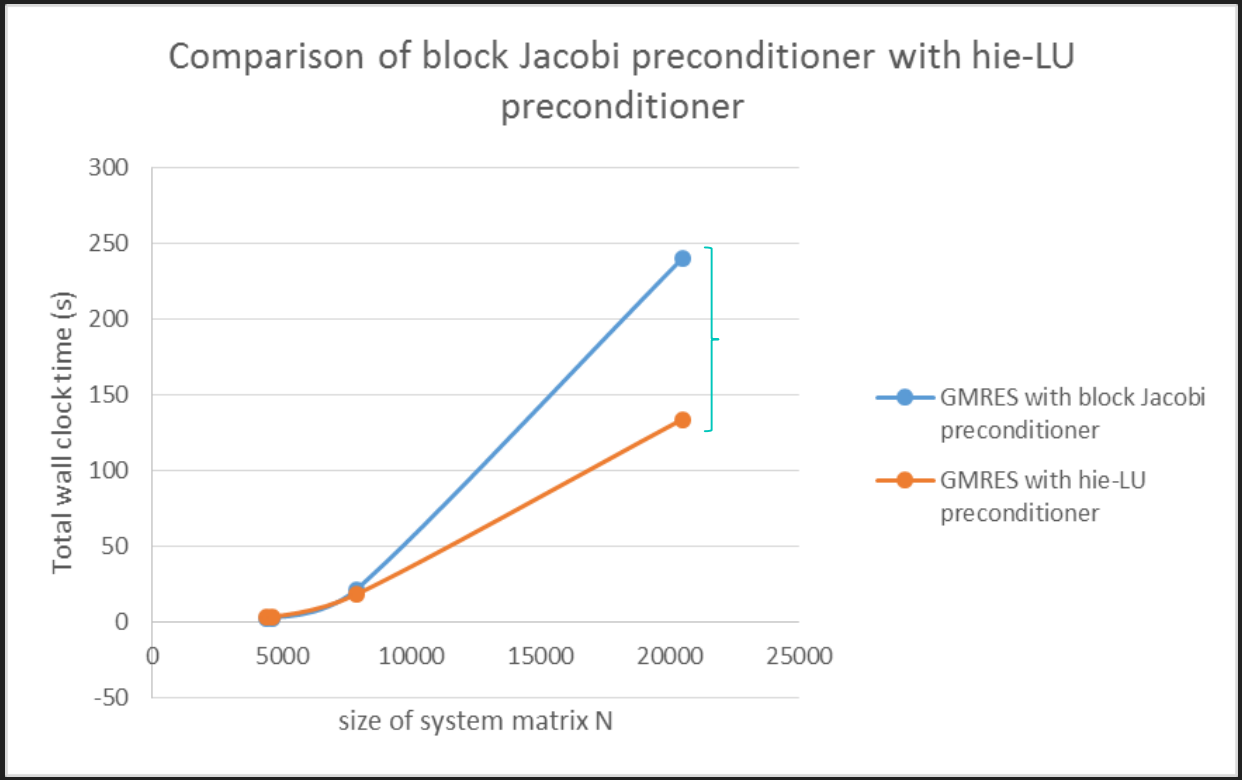
Hierarchical Matrix Arithmetic have to be defined

Integration with solver

$$(LU)^{-1}Ax = (LU)^{-1}b$$

- First apply hierarchical lower triangular solver to obtain $L^{-1}b$
- Then apply hierarchical upper triangular solver to obtain $U^{-1}L^{-1}b$
- Same goes for vector Ax

Results in Serial



- Time required to solve the largest FATIMA_20493 is 44% less
- Block Jacobi scales with $O\left(\left(\frac{N}{number\ of\ blocks}\right)^3\right)$
- Hie-LU scales with $O(N(logN)^2)$
- Time saved is expected to increase with increase in problem size

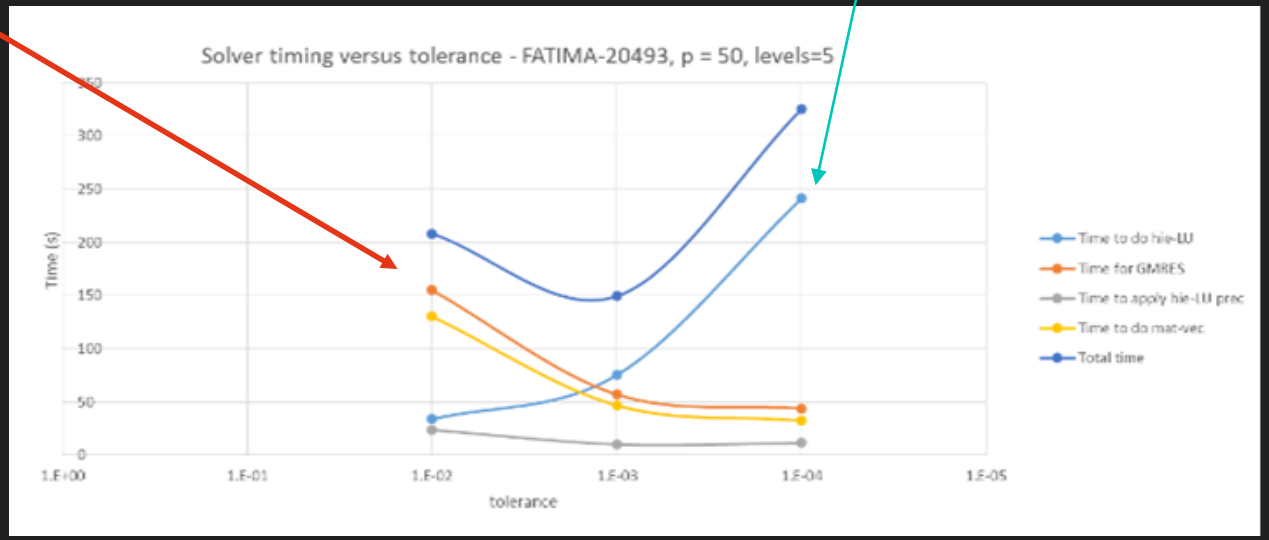
TU Delft

Results in Serial

	Results using block Jacobi Preconditioner					Results using hie-LU preconditioner				
Matrix	Block-size	GMRES		IDR(s)		Variables for hie_LU	GMRES		IDR(s)	
		Wall clock time (s)	# iters	Wall clock time (s)	# iters		Wall clock time (s)	# iters	Wall clock time (s)	# iters
FATIMA_20493	4000	239.8	103	249.836	110	Tol: 1e-03 b = 200 p = 50	134.74	51	140.35	55
FATIMA_7894	1000	21.3	121	23.23	133	Tol: 1e-03 b = 200 p = 50	18.72	11	19.03	11
PASSCAL	500	2.2	91	2.7	106	Tol: 1e-02 b = 100 p = 20	2.62	44	2.72	46
STEADYCAV1	500	1.7	61	1.9	68	Tol: 1e-02 b = 100 p = 30	2.65	32	2.86	37

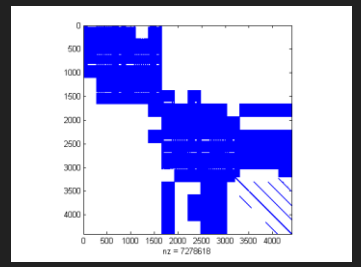
Influence of tolerance tol-hie

- Tol_hie controls how accurate the low rank approximation is to each matrix block
  - The lower it is, the more the number of admissible blocks
  - But the less accurate it is, hence, the more the number of iterations required for convergence



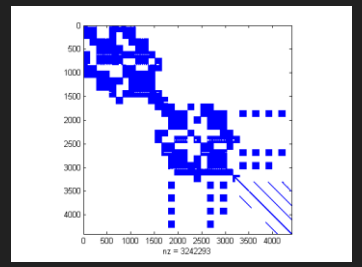
Influence of minimum block size b

- The minimum block size b controls the depth of recursion
  - An additional level of recursion implies additional work and storage
  - But it also implies more blocks become admissible



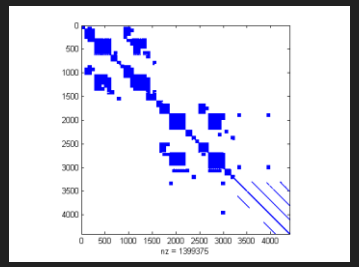
Levels = 4

Hie-LU time = 4.89s



Levels = 5

Hie-LU time = 4.02s



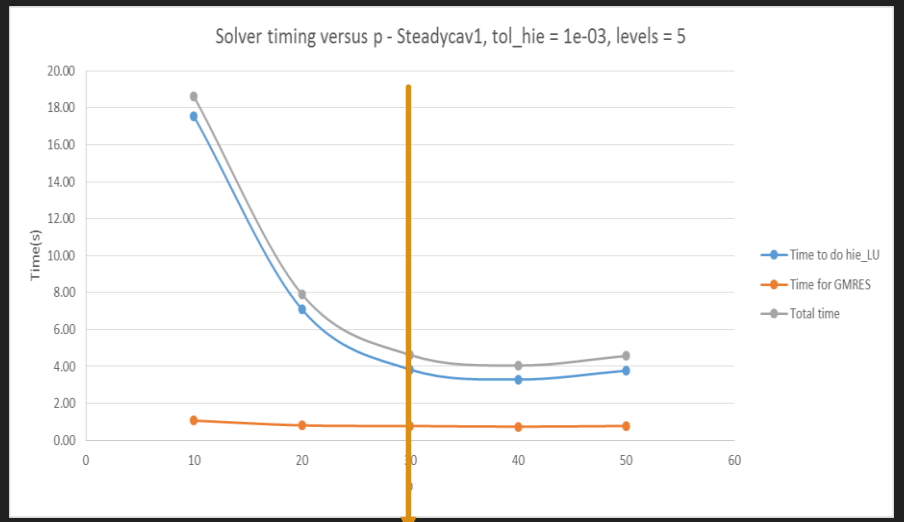
Levels = 6

Hie-LU time = 5.64s

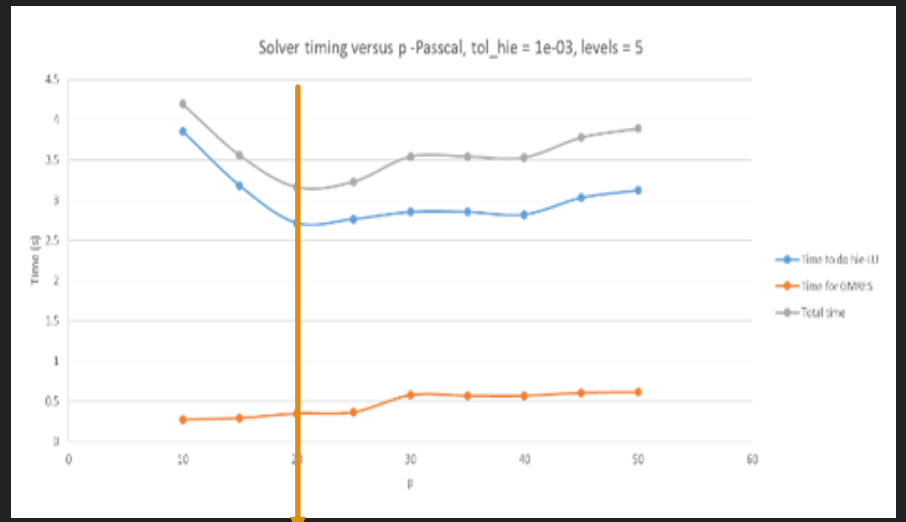
Passcal, p = 35, tol_hie = 1e-4

Influence of rank of low rank approximation p

- Value of p is very dependent on the inherent rank of the off diagonal blocks
  - For example, Steadycav and PASSCAL shown
  - As p increases beyond threshold p, time could increase or decrease slightly



Threshold p = 30



Threshold p = 20

Parallel implementation

- Because block Jacobi's main benefit is its parallelizability, it is of interest to compare the parallel performance
- Hie-LU is parallelized as follow
- Other routines are also parallelized using the "Sections" construct

```
1)  If l =finest recursion level
    a)  If adm_A = 0
        i)   Dense matrix multiplication (XGEMM)
    b)  Else
        i)  Low rank matrix multiplication (XGEMM)
2)  Else
    a)  If adm_A = 1
        i)  Low rank matrix multiplication (XGEMM)
    b)  Else, split the block into four sub-blocks, and M_in and M_out into two
        halves according to Equation 12.
        PARALLEL SECTIONS
        SECTION 1:
        i)  M_out_1 = M_out_1 + A_11 M_in_1 + A_12 M_in_2
        SECTION 2:
        ii) M_out_2 = M_out_2 + A_21 M_in_1 + M_22 M_in_2
        END PARALLEL SECTIONS
```

TUDelft

Results in parallel

		Results using block Jacobi preconditioner					Results using hie-LU preconditioner				
Matrix	nrhs	Block-size	GMRES		IDR(s)		Variables for hie_LU	GMRES		IDR(s)		
			Wall clock time (s)	# iters	Wall clock time (s)	# iters		Wall clock time (s)	# iters	Wall clock time (s)	# iters	
FATIMA_20493	1	4000 for GMRES 1708 for IDR(s)	87.62	103	80.45	260	Tol: 1e-03 b = 200 p = 50	47.48	51	50.14	55	40 % savings
	7	4000	211.49	103	220.48	112	Tol: 1e-04 b = 100 p = 30	108.96	30	116.20	32	48 % savings
FATIMA_7894	1	1000	6.36	121	6.55	133	Tol: 1e-03 b = 200 p = 50	6.44	11	6.57	11	
	7	1000	25.74	121	28.18	136	Tol: 1e-03 b = 200 p = 40	9.17	11	9.74	11	65 % savings
PASSCAL	1	500	0.72	91	0.76	96	Tol: 1e-03 b = 100 p = 20	1.42	11	1.49	11	
STEADYCAV1	1	500	0.57	61	0.58	62	Tol: 1e-02 b = 100 p = 30	1.934	32	2.12	37	

Conclusion

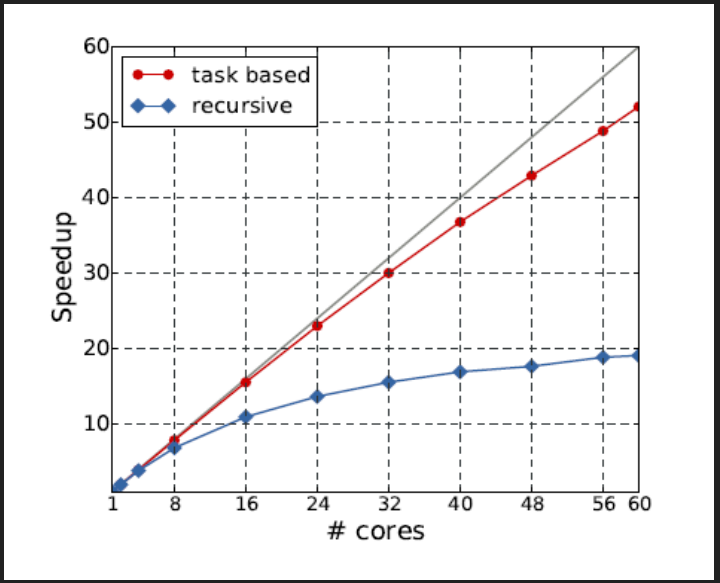
Conclusion

Final strategy recommended:

IDR(s) with hierarchical LU preconditioner

Recommendations

- Better parallelization strategy
  - Task based parallelization
    - Work by Ronald Kriemann has shown that optimal speedup with good scaling behaviour is possible using task-based parallelization strategy
  - Use of GPUs
    - The most expensive parts of the hie-LU algorithm is at its leaves, where dense matrix operations need to be carried out
    - If MAGMA library can be used, it could reduce the time significantly
- To reduce the complexity further, consider Fast Multipole Method or H2 matrix



Extracted from: Kriemaan, R. 2014. H-LU Factorization on Many-Core Systems. Computing and Visualization in Science, Vol 16, Issue 3.

The End

Thank you very much!

TU Delft