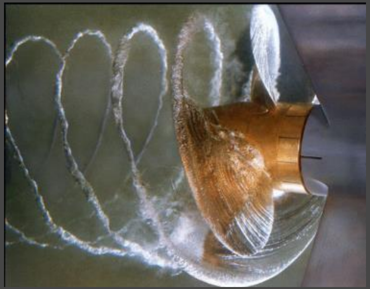
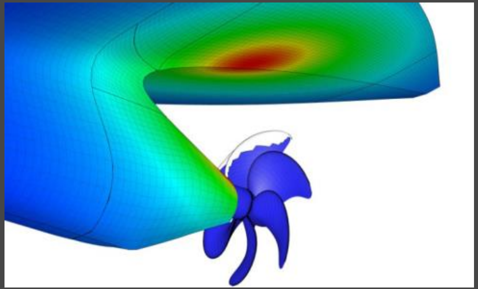
 

Extracted from: http://www.marin.nl

EFFICIENCY IMPROVEMENT FOR PANEL CODES

LITERATURE REVIEW

Elisa Ang

30 Jan 2015, Friday

Project Overview

- Panel codes (or Boundary Element Method) are used for flow computations in MARIN
- Boundary Element Method generates dense linear system of equation
- The project aims to speed up the computation time required

Presentation Topics

- Review of what had been done
- Boundary Element Method
- Solver Methods comparison:

  GMRES vs IDR(s)
- Preconditioning:

  Block Jacobi vs Deflation
- Fast Multipole Method
- Subsequent plan

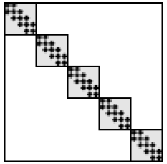
Introduction

- Current Strategy in MARIN:
  - Direct Solver or GMRES with incomplete LU preconditioner

Matrix Name	Size	Real/Complex	Strategy	Solve time
Steadycav	4620	Real	Direct	3.3 s (direct)
FATIMA_7894	7894	Complex	ILU	13.0 s
FATIMA_20493	20493	Complex	ILU	170.0 s

Review of what had been done

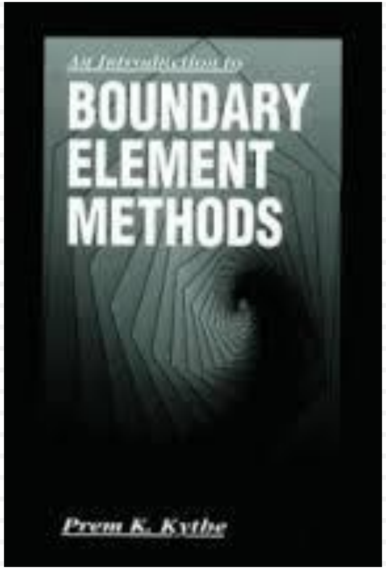
- Project was undertaken by Martijn de Jong in 2012
- The solution was GMRES with Block-Jacobi Preconditioner with OpenMP
- GPU was used to speed up the solver



Matrix Name	Size	Real/ Complex	Solve time (Old)	Solve time (Now)	Solve time (GPU)
Steadycav	4620	Real	3.3 s (Direct)	0.5s	Not tested
FATIMA_7894	7894	Complex	13.0s (iLU)	4.5s	3.1s (Direct)
FATIMA_20493	20493	Complex	170.0 s (iLU)	34.3s	22.7s (Block-Jac)

All results and diagrams extracted from:
de Jong, M. 2012. *Efficient Solvers For Panel Codes*

Boundary Element Method



Kythe, K.P. 1995. An Introduction to Boundary Element Methods

Boundary Element Method (BEM)

- Numerical method to solve boundary value problems

$$-\nabla^2 u = 0$$

- FEM vs BEM
  - FEM solves this by discretizing entire domain, BEM only discretizes boundary
  - FEM results in sparse matrix, BEM, dense

Boundary Element Method (BEM)

Equation

$$-\nabla^2 u = 0$$

$$u = u_0 \ on \ S_1$$

$$q = \frac{\partial u}{\partial n} = q_0 \ on \ S_2$$

- Weak Formulation

$$-\iiint_V u^* \nabla^2 u \, dV = \iiint_V \nabla u^* \nabla u \, dV - \iint_S u^* \nabla \mathrm{u} \cdot \hat{\boldsymbol{n}} \, ds = 0$$

$$-\iiint_V u \nabla^2 u^* \, dV + \iint_S u \nabla \mathrm{u}^* \cdot \hat{\boldsymbol{n}} \, ds - \iint_S u^* \nabla \mathrm{u} \cdot \hat{\boldsymbol{n}} \, ds = 0$$

- u^* = Fundamental Solution

$$\nabla^2 u^* = -\delta(x_i) \qquad u^* = \frac{1}{4\pi(x - x_i)}$$

Boundary Element Method (BEM)

Equation

$$\nabla^2 u = 0$$

$$u = u_0 \; on \; S_1$$

$$q = \frac{\partial u}{\partial n} = q_0 \; on \; S_2$$

- Thus we have the Boundary Integral Equation (BIE)

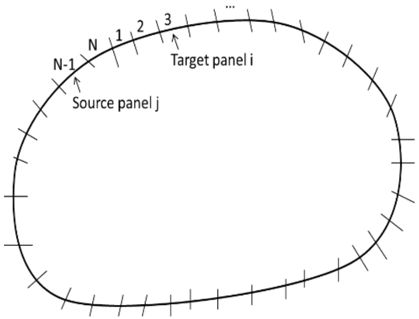
$$c(x_i)u(x_i) + \iint_S uq_i^* \, dS = \iint_S u_i^* q \, dS \,, \qquad S = S_1 \cup S_2$$

$$c(x_i) = \begin{cases} 1 & if \; x_i \; is \; inside \; R \\ \frac{1}{2} & if \; x_i \; is \; on \; a \; smooth \; portion \; of \; S \end{cases}$$

Boundary Element Method (BEM)

- Discretization of BIE gives:

$$c(x_i)u(x_i) + \iint_S uq_i^*\, dS = \iint_S u_i^* q\, dS\,,$$



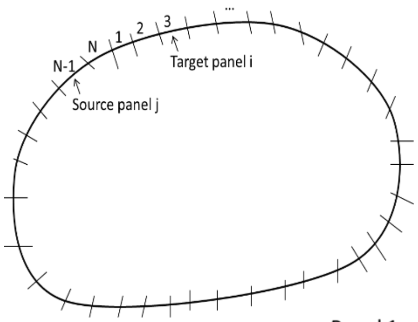
Boundary Element Method (BEM)

- Discretization of BIE gives:

$$c(x_i)u(x_i) + \sum_{j=1}^{N} u(x_j)\,\widehat{H}_{ij} = \sum_{j=1}^{N} q(x_j)\,G_{ij}$$

$$\widehat{H}_{ij} = \iint_{S_j} q_i^*\,dS \qquad q^* = -\frac{1}{4\pi(x-x_i)^{2}}$$

$$G_{ij} = \iint_{S_j} u_i^*\,dS \qquad u^* = \frac{1}{4\pi(x-x_i)}$$



$$\begin{array}{c|c}
 & \begin{array}{cccccc} \text{Panel 1} & \text{Panel 2} & \ldots & \text{Panel j} & \ldots & \text{Panel N} \end{array} \\
\begin{array}{c} \text{Panel 1} \\ \text{Panel 2} \\ \vdots \\ \text{Panel i} \\ \vdots \\ \text{Panel N} \end{array} &
\begin{pmatrix} a(1,1) & a(1,2) & \ldots & a(1,N) \\ & \ddots & & \\ \vdots & & a(i,j) & \vdots \\ & & \ddots & \\ a(N,1) & a(N,2) & \ldots & a(1,N) \end{pmatrix}
\end{array}$$

- Rearranging -> Dense linear system $Ax=b$

Boundary Element Method

- BEM test matrices from MARIN

Name	Size	Real/Complex	System
Steadycav1	4620	Real	PROCAL
Steadycav2	4620	Real	PROCAL
Steadycav3	4620	Real	PROCAL
Steadycav4	4649	Real	PROCAL
DIFFRAC_1828	1828	Complex	DIFFRAC
FATIMA_7894	7894	Complex	FATIMA
FATIMA_20493	20493	Complex	FATIMA

Solver

GMRES vs IDR(s)

www.allacronyms.com

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

Solver Method

- Advantage of GMRES:
  - Optimality
  - matrix vector multiplication required per iteration
- Advantage of IDR(s):
  - short recurrence
  - less matrix vector multiplication required as compared to bi-CG

Solver Method

- GMRES
  - Search for solution within increasing Krylov Subspace
- IDR(s)
  - Concept of nested subspace

$$\mathcal{G}_j \subset \mathcal{G}_{j-1} \quad \text{with} \quad \mathcal{G}_0 = \mathcal{K}(A, r_0)$$

$$\mathcal{G}_j = (\boldsymbol{I} - \omega_j \boldsymbol{A})(\mathcal{G}_{j-1} \cap \mathcal{S})$$

Scalar value

left null space of some N x s matrix P

$$\mathcal{G}_j = \{\mathbf{0}\}, for\ some\ j \leq N$$

Solver Method

Numerical Results

	Matrix	GMRES	IDR(10)	Bi-CG
Timing (s)	Steadycav1	5.4	5.9	16
Iteration	N=4620	237	379	490

	Matrix	GMRES with block jacobi preconditioned matrix	IDR(10) with block jacobi preconditioned matrix
Timing for solving (s)	FATIMA_20493	1948	894.3
Iteration		96	115

Preconditioner

Preconditioner

- Martijn did a thorough comparison between ILU and Block Jacobi
- Can deflation further reduce the iterations required for convergence?

Preconditioner

- Short review of Deflation
  - Split solution space into 2 complementary subspaces through projection

$$x = (I - Q_D)x + Q_D x$$

  - Define the projectors

$$P_D = I - AZE^{-1}Y^T$$

$$Q_D = I - ZE^{-1}Y^T A$$

Deflation Subspaces

$Z, Y \in \mathbb{C}^{nxr}$

$E = Y^T AZ$

Preconditioner

$$x = (I - Q_D)x + Q_D x$$

$$P_D = I - AZE^{-1}Y^T$$

$$Q_D = I - ZE^{-1}Y^T A$$

- Consider
  - $Ax = (I - P_D)Ax + P_D Ax$

    $$P_D Ax = P_D b$$

    $\tilde{x}$
  - $x = ZE^{-1}Y^T b + Q_D \tilde{x}$

Preconditioner

$$x = (I - Q_D)x + Q_D x$$

$$P_D = I - AZE^{-1}Y^T$$

$$Q_D = I - ZE^{-1}Y^T A$$

$$\downarrow$$

$$P_D A x = P_D b$$

- Choice of deflation subspace Z and Y
  - Space spanned by eigenvectors of A corresponding to smallest eigenvalues
    - Effect is to shift the small eigenvalues to 0 while leaving the other eigenvalues unchanged

Preconditioner

$$x = (I - Q_D)x + Q_D x$$

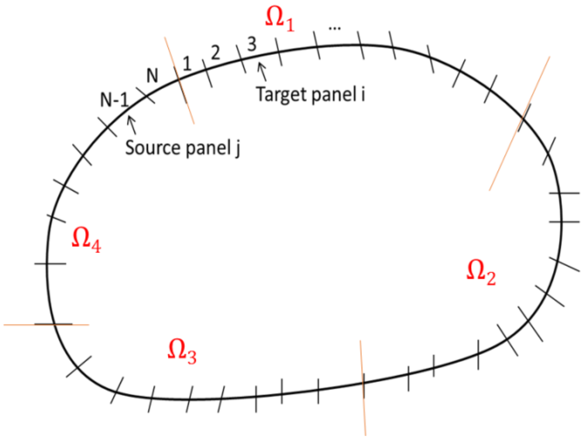
$$P_D = I - AZE^{-1}Y^T$$

$$Q_D = I - ZE^{-1}Y^T A$$

$$\downarrow$$

$$P_D A x = P_D b$$

- Choice of deflation subspace Z and Y
  - Subdomain decomposition



$$Z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & \vdots & \vdots \\ 0 & 1 & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & 1 & 0 & \vdots \\ \vdots & 0 & 1 & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & 1 & \vdots \\ \vdots & \vdots & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Preconditioner

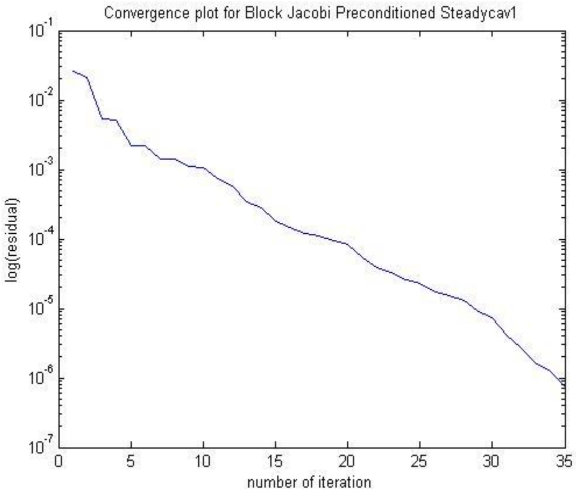
- Numerical Results

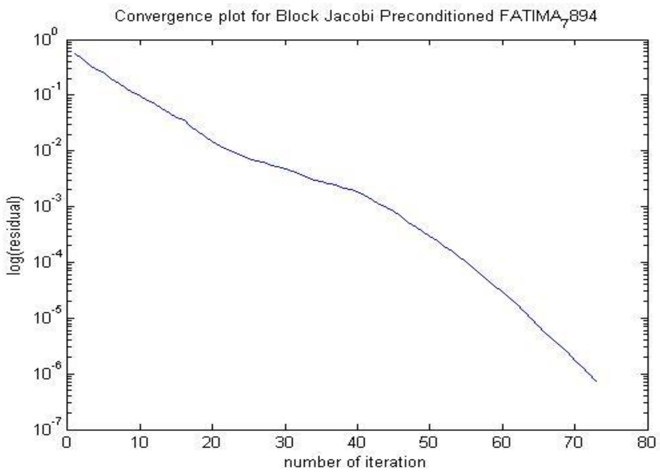
	Matrix	IDR(10) with block jacobi	IDR(10) with subdomain deflation
Timing for preconditioning (s)	Steadycav1 N=4620	0.65	0.69 BJ 0.13 D
Timing for solving (s)		0.49	0.44
Total time (s)		1.13	1.26
Iteration		45	41
Timing for preconditioning (s)	FATIMA 7894 N=7894	7.7	8.0 BJ 1.1 D
Timing for solving (s)		10.1	10.2
Total time (s)		17.8	19.3
Iteration		84	86

No Improvement

Preconditioner

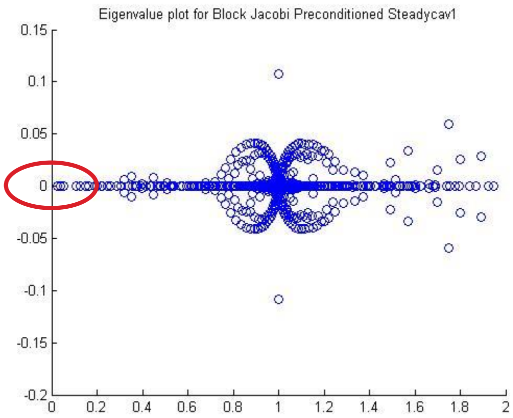
- Further analysis of why deflation do not provide any improvement

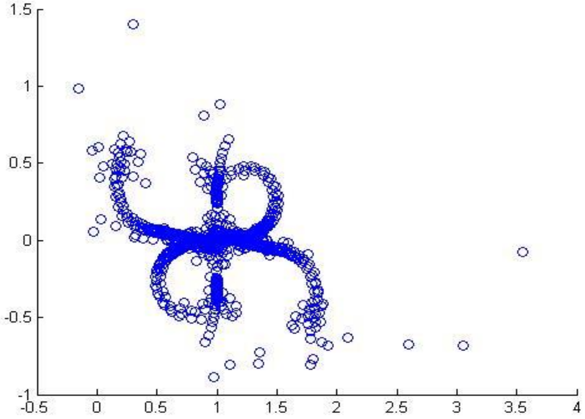




Preconditioner

- Further analysis of why deflation do not provide any improvement





Preconditioner

Numerical Results

	Matrix	IDR(10) with block jacobi	IDR(10) with subdomain deflation	IDR(10) with eigenvectors deflation
Timing for preconditioning (s)	Steadycav1 N=4620	0.65	0.69 BJ 0.13 D	0.69 BJ 2.25 D
Timing for solving (s)		0.49	0.44	0.29
Total time (s)		1.13	1.26	3.23
Iteration		45	41	24
Timing for preconditioning (s)	FATIMA 7894 N=7894	7.7	8.0 BJ 1.1 D	8.0BJ 52.5D
Timing for solving (s)		10.1	10.2	7.8
Total time (s)		17.8	19.3	68.3
Iteration		84	86	66

But there's a need to improve the time to construct the deflation space

Reduction!

Fast Multipole Method (FMM)

Leslie Greengard
New York University

Vladimir Rokhlin
Yale University

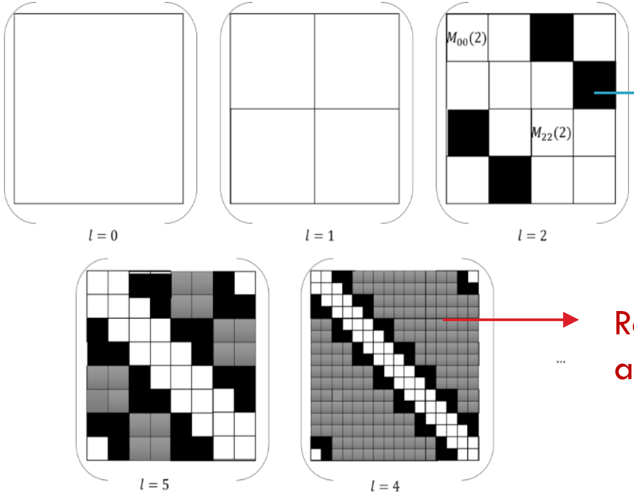
1996

Fast Multipole Method (FMM)

- Speed up matrix-vector multiplication from $O(N^2)$ to $O(N\log N)$ or $O(N)$
- Exploit the hierarchical structure of the matrix
- What is the hierarchical structure?

Fast Multipole Method (FMM)

- Hierarchical Structure
  - First, this is an example of the hierarchical splitting of a matrix A



Represents an admissible block:
An admissible block has **low rank**

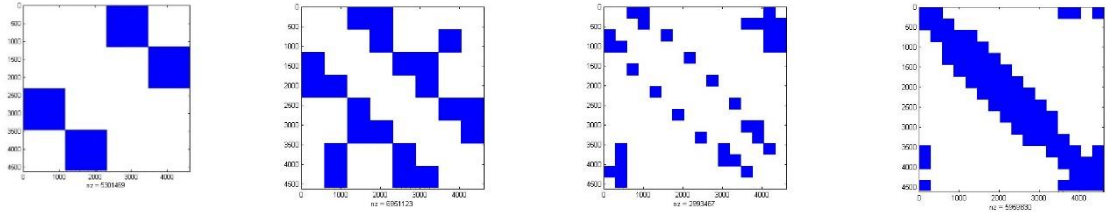
Represents previously admissible block

$$A = \sum_{l=2}^{n} M_l + N_n$$

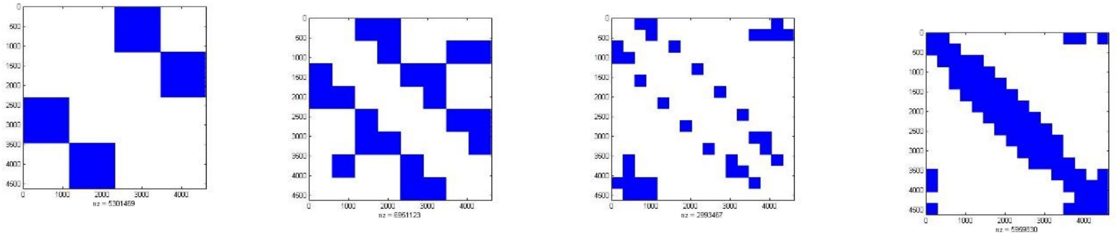
Fast Multipole Method (FMM)

- Hierarchical Structure
  - All test matrices have the same hierarchical structure

Steadycav1

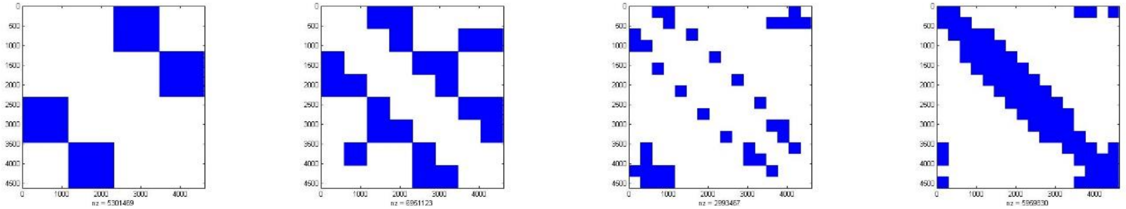


Stedycav2

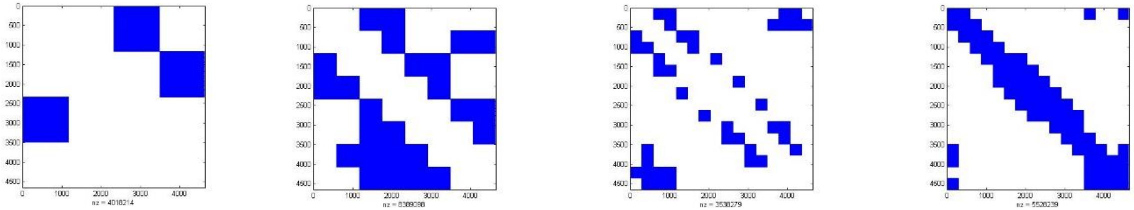


Fast Multipole Method (FMM)

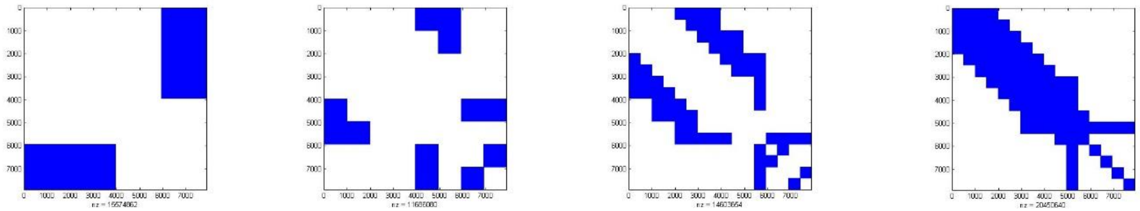
Stedycav3



Stedycav4



FATIMA_7894

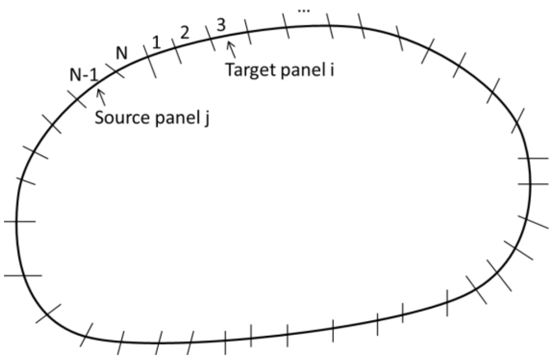


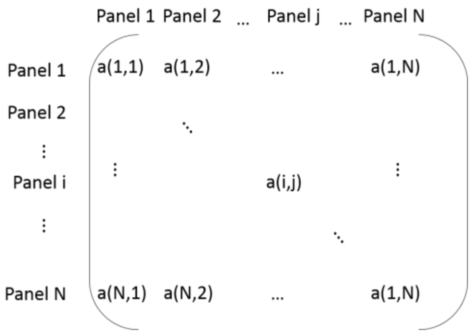
Fast Multipole Method (FMM)

- Why is the blocks of low rank?
- How can we use the fact that the blocks are low rank?

Fast Multipole Method (FMM)

- Recall that each element of our matrix A is:





$$a(x_i, x_j) = \int_{S_j} q^*(x_i, x_j)dS \;\; or \;\; \int_{S_j} u^*(x_i, x_j)dS$$

$$q^* = -\frac{1}{4\pi(x - x_i)^2}$$

$$u^* = \frac{1}{4\pi(x - x_i)}$$

Fast Multipole Method (FMM)

- Let K(x,y)=a$\left(x_i, x_j\right)$
- And apply Taylor Expansion, centred around $\left(c_\sigma, c_\tau\right)$

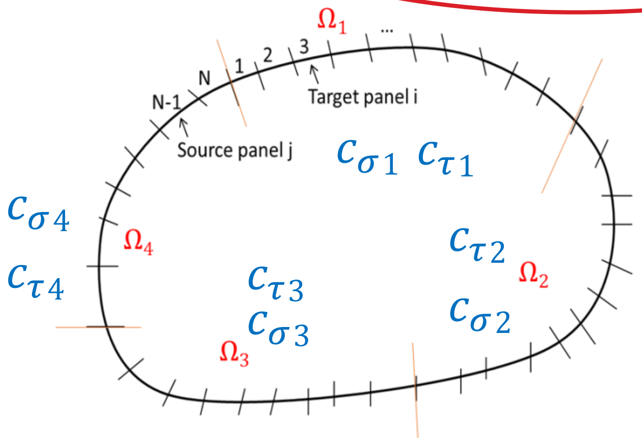
$$K(x,y) = \sum_{l=0}^{p-1} \frac{1}{l!} \left[(x - c_\sigma)\partial_x + (y - c_\tau)\partial_y\right]^l K(c_\sigma - c_\tau) + R_p(x,y)$$

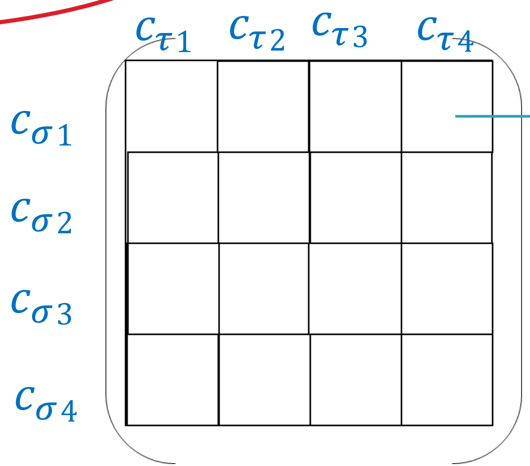
$$\approx 0\ if \left|\frac{(x - c_\sigma) + (y - c_\tau)}{|x - y|}\right| < 1$$

Fast Multipole Method (FMM)

- Applying binomial expansion and simplifying:

$$K(x,y) = \sum_{m=0}^{l+m} \sum_{l=-m}^{p-1-m} \frac{1}{m!\,l!} \partial_x^l \partial_y^m K(c_\sigma - c_\tau)\,(x - c_\sigma)^l (y - c_\tau)^m$$





All elements within each block has the same (c_σ, c_τ)

Fast Multipole Method (FMM)

- Applying binomial expansion and simplifying:

$$K(x,y) = \sum_{m=0}^{l+m} \sum_{l=-m}^{p-1-m} \frac{1}{m!\,l!} \partial_x^l \partial_y^m K(c_\sigma - c_\tau)\,(x - c_\sigma)^l (y - c_\tau)^m$$

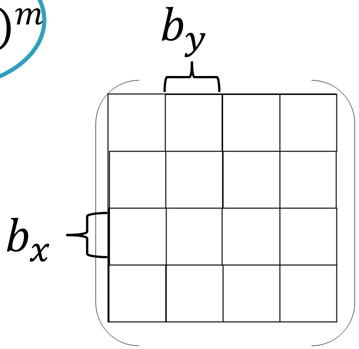
Thus for each block, we can define upper triangular matrix $S^{\sigma,\tau} \in \mathbb{C}^{pxp}$

$$s_{l,m} = \begin{cases} \frac{1}{l!\,m!} \partial_x^l \partial_y^m K(c_\sigma, c_\tau) \quad if & if\ 0 \le l+m \le p-1 \\ 0 & else \end{cases}$$

Fast Multipole Method (FMM)

- Applying binomial expansion and simplifying:

$$K(x,y) = \sum_{m=0}^{l+m} \sum_{l=-m}^{p-1-m} \frac{1}{m!\,l!} \partial_x^l \partial_y^m K(c_\sigma - c_\tau)(x - c_\sigma)^l (y - c_\tau)^m$$



Then we define 2 matrices, $\Psi^\tau \in \mathbb{C}^{b_y x p}, \Psi^\sigma \in \mathbb{C}^{b_x x p}$

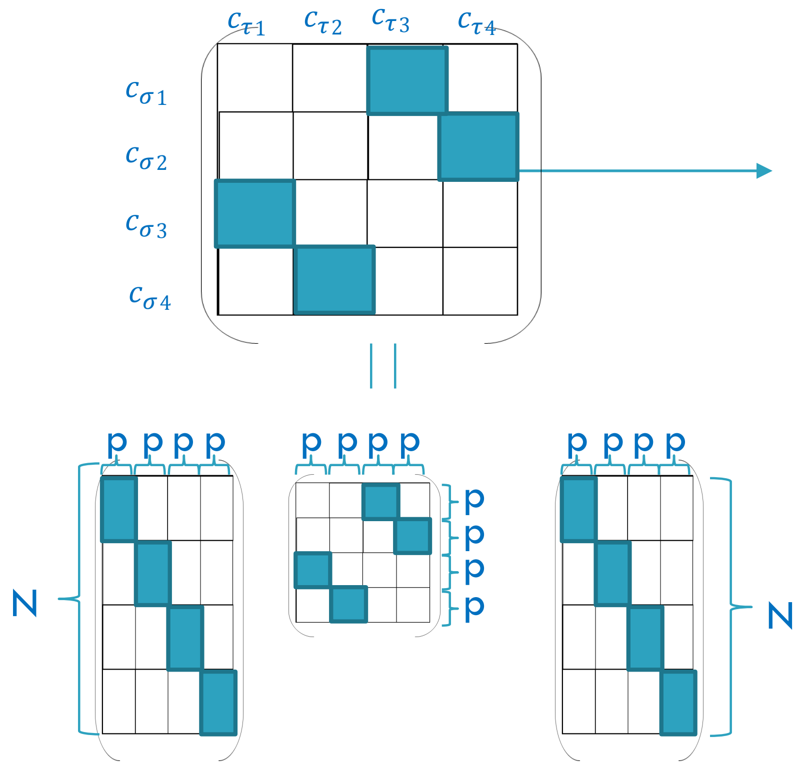
$$\psi^\sigma_{i\times l} = (x - c_\sigma)^l, \qquad x = X(i),$$

$$where \ \frac{\sigma N}{2^l} \leq i \leq \frac{\sigma N}{2^l} + b$$

$$\psi^\tau_{j\times m} = (y - c_\tau)^m, \qquad y = X(j),$$

$$where \ \frac{\sigma N}{2^l} \leq j \leq \frac{\sigma N}{2^l} + b$$

Fast Multipole Method (FMM)



Each admissible block can be written as:

$$M_{\sigma,\tau}(l) \approx \widetilde{M}_{\sigma,\tau}(l) = (\Psi^{\sigma}) S^{\sigma,\tau} (\Psi^{\tau})^T$$

Each block in the same row has the same Ψ^{σ}, and in the same col with the same Ψ^{τ}

$$M_l \approx \sum \widetilde{M}_{\sigma,\tau}(l)$$

$$= \underbrace{blockdiag(\Psi^{\sigma})}_{N \times p2^l}{}_{\sigma=0,1,\ldots,2^l} \underbrace{S(l)}_{p2^l \times p2^l} \underbrace{blockdiag(\Psi^{\tau})^T}_{N \times p2^l}{}_{\tau=0,1,\ldots,2^l}$$

Fast Multipole Method (FMM)

- Consider now the matrix vector multiplication Ax

$$Ax \approx \sum_{l=2}^{n} \widetilde{M}_l x + N_n x$$

$$\underbrace{M_l x = blockdiag(\Psi^{\sigma})_{\sigma=0,1,\dots,2^l} \underbrace{S(l) \underbrace{blockdiag(\Psi^{\tau})^T{}_{\tau=0,1,\dots,2^l}\, x}_{N \times p2^l}}_{p2^l \times p2^l}}_{N \times p2^l} \quad \Bigg\} \; O(N)$$

Fast Multipole Method (FMM)

- Consider now the matrix vector multiplication Ax

$$Ax \approx \sum_{l=2}^{n} \widetilde{M}_l x + N_n x$$

It can be shown that N_n is a sparse matrix with at most cxN non zero elements $\Big\}$ O(N)

Fast Multipole Method (FMM)

- Consider now the matrix vector multiplication Ax

$$Ax \approx \sum_{l=2}^{n} \widetilde{M}_l x + N_n x$$

Ax is now a O(N) operation

Fast Multipole Method (FMM)

- Other ways to obtain low rank approximation

$$M_{\sigma,\tau}(l) \approx \widetilde{M}_{\sigma,\tau}(l) = (\Psi^{\sigma}) S^{\sigma,\tau} (\Psi^{\tau})^{T}$$

- Without domain & kernel information, we can use lanzcos bidiagonalization to check for admissibility and obtain low rank approximation

$$M_{\sigma,\tau}(l) \approx \widetilde{M}_{\sigma,\tau}(l) = U_{\sigma,\tau} B_{\sigma,\tau} V_{\sigma,\tau}^{H}$$

Fast Multipole Method (FMM)

- But we can’t form block diag matrix for $M(l)$
- Matrix vector multiplication has to be done like this:

$$\widetilde{M}_l x = \begin{bmatrix} \left(\widetilde{M}_l x\right)_1 \\ \vdots \\ \left(\widetilde{M}_l x\right)_{2^l} \end{bmatrix} = \begin{bmatrix} \sum_{\tau=1}^{2^l} \widehat{U}_{1,\tau} B_{1,\tau} \widehat{V}_{1,\tau}^H x_\tau \\ \vdots \\ \sum_{\tau=1}^{2^l} \widehat{U}_{2^l,\tau} B_{2^l,\tau} \widehat{V}_{2^l,\tau}^H x_\tau \end{bmatrix}$$

- O(NlogN)

Fast Multipole Method (FMM)

- Numerical Results

		Steadycav1				FATIMA_7894			
b	p	t_{m-full}	t_{m-hie}	% reduction	t_{split}	t_{m-full}	t_{m-hie}	% reduction	t_{split}
100	10	0.043	0.034	20.93%	1.35	0.3	0.14	53.33%	11.03
	20		0.028	34.88%	2.22		0.12	60.00%	18.7
	35		0.024	44.19%	3.79		0.13	56.67%	31.46
	50		0.024	44.19%	5.07		0.12	60.00%	48.6
200	10	0.034	0.032	5.88%	0.91		0.15	40.00%	9.4
	20		0.0286	15.88%	1.5		0.13	48.00%	15.01
	35		0.02	41.18%	2.34		0.11	56.00%	24.33
	50		0.018	47.06%	3.32	0.25	0.11	56.00%	33.25

Fast Multipole Method (FMM)

- Numerical Results – Storage Requirement

	Steadycav1	Steadycav2	Steadycav3	Steadycav4	FATIMA_7894
Storage requirement for full matrix	**0.17 GB**	**0.17 GB**	**0.17 GB**	**0.17 GB**	**0.99 GB**
Storage requirement in hierarchical form (b=100, p=50)	N: 0.086 GB B: 3.71e-4 GB U: 0.038 GB V: 0.037 GB **Total: 0.16 GB**	N: 0.085 GB B: 3.74e-4 GB U: 0.038 GB V: 0.037 GB **Total: 0.16 GB**	N: 0.084 GB B: 3.81e-4 GB U: 0.038 GB V: 0.038 GB **Total: 0.16 GB**	N: 0.08 GB B: 3.82e-4 GB U: 0.04 GB V: 0.039 GB **Total: 0.16 GB**	N: 0.56 GB B: 5.28e-4 GB U: 0.078 GB V: 0.077 GB **Total: 0.72 GB**
Decrease in storage required	5.88%	5.88%	5.88%	5.88%	27.3%

Conclusion

" We're under a lot of time-pressure here, so we'll need to jump to conclusions. "

Conclusion &Subsequent Plan

- Implement IDR(s) in place of GMRES
- Explore efficient implementation of deflation with eigenvectors
- Explore use of GPU
- Fast Multipole Method with lanzcos bidiagonalization
- Fast Multipole Method with domain and kernel information

Thank you!