A fourth order finite difference method for solving elliptic interface problems with the FFT acceleration

Shan Zhao

Department of Mathematics,
University of Alabama, USA

Conference on Fast Direct Solvers

October 24, 2021

Collaborators

Dr. Emil Alexov (Clemson Univ. USA)

Dr. Weihua Geng (Southern Methodist Univ. USA)

Dr. Chuan Li, (West Chester Univ. of Pennsylvania, USA)

Dr. Lunji Song, (Lanzhou Univ., China)

Dr. Guangqing Long, (Nanning Normal Univ., China)

PhD students (Univ. of Alabama)

Dr. Hongsong Feng (Postdoc, Michigan State Univ.)

Yimin Ren, Mark McGowan, Sylvia Amihere.

Funding support

NSF DMS-1812930

NSF DMS-2110914

Outline

1. **Elliptic interface problem**

2. **Poisson boundary value problem (BVP)**
   - 4^{th} - 8^{th} order for 2D & 3D cubic domains
   - FFT for any boundary condition

3. **Poisson interface problem**
   - Fourth order accuracy for curved interfaces
   - FFT for both interface and boundary

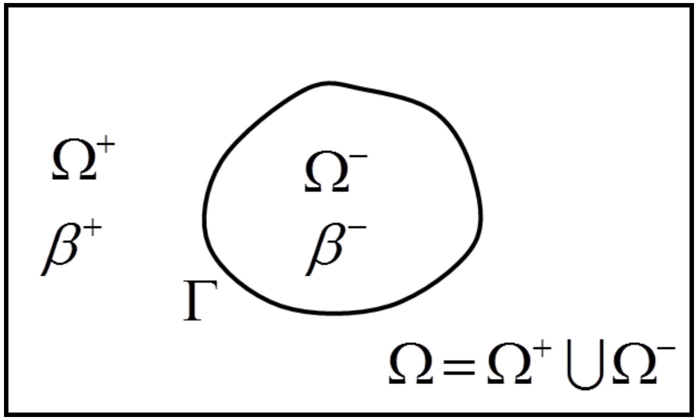
4. **Summary**

1. Elliptic interface problem

Poisson's equation over a rectangular domain

$$\begin{aligned} -\nabla \cdot (\beta \nabla u) &= f(\vec{x}), \qquad \vec{x} \in \Omega \\ u &= g(\vec{x}), \qquad \vec{x} \in \partial\Omega \end{aligned} \qquad (1)$$

where $\beta = \begin{cases} \beta^+, & \vec{x} \in \Omega^+ \\ \beta^-, & \vec{x} \in \Omega^- \end{cases}$

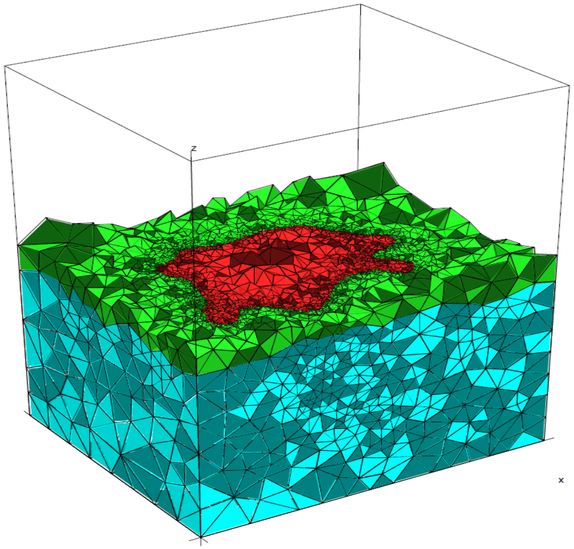


- **General jump conditions**

$$\begin{aligned} [u] &:= u^+ - u^- = \phi(\vec{x}) \\ [\beta u_n] &:= \beta^+ \nabla u^+ \cdot \vec{n} - \beta^- \nabla u^- \cdot \vec{n} = \psi(\vec{x}) \end{aligned} \qquad (2)$$

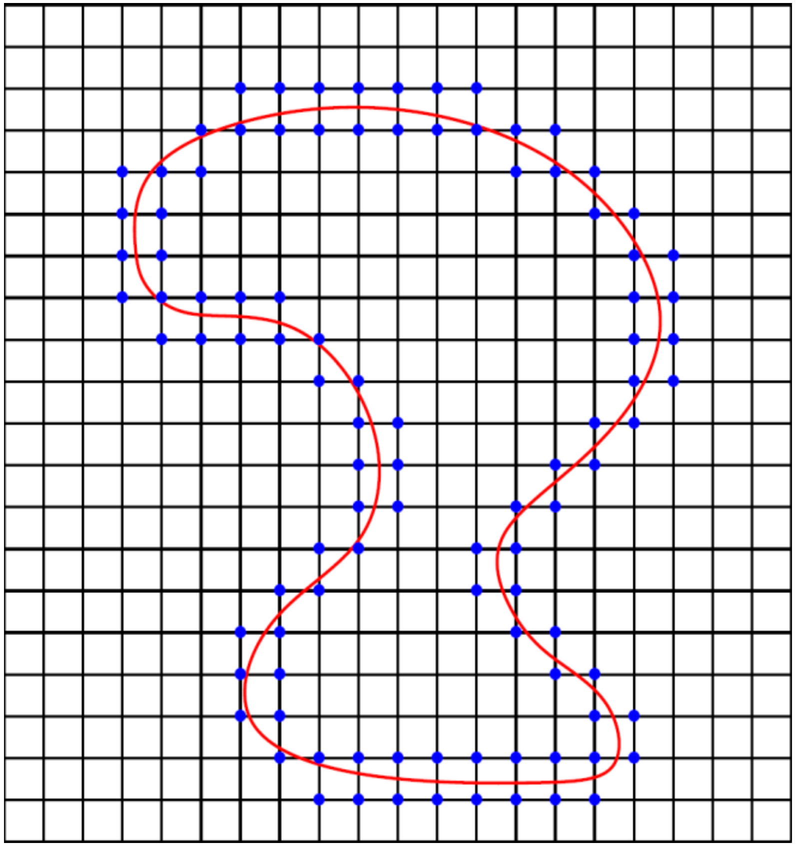
Body-fitted meshes

- **Finite element methods (FEMs)**
  - Pioneer studies *(Babuska 1970; Chen, Zou 1998)*
  - Immersed FEM (IFEM) *(Ewing, Li, Lin, Lin 1999)*
  - Extended FEM (XFEM) *(Ji, Dolbow 2004)*
  - Discontinuous Galerkin *(Huynh et. 2013)*
  - Weak Galerkin *(Mu et. 2013)*
  - …
- **Finite volume methods**
- **Integral equation methods**



Cartesian grid methods

- Immersed boundary method *(Peskin 1977)*
- Immersed interface method (IIM) *(LeVeque, Li, 1994)*
- Ghost fluid method *(Fedkiw, Osher, Liu 2000)*
- Matched interface and boundary (MIB) *(Zhou, Zhao, Wei 2006)*
- …



Fast Poisson solvers for interface problems

- Geometric multigrid solver O(N)
  - IIM *(Adams, Li, 2004)*
  - Piecewise-polynomial interface method *(Chen, Strang, 2008)*
  - Ghost fluid method *(Coco, Russo, 2018)*
  - Virtual node method *(Bedrossian, et. 2010)*

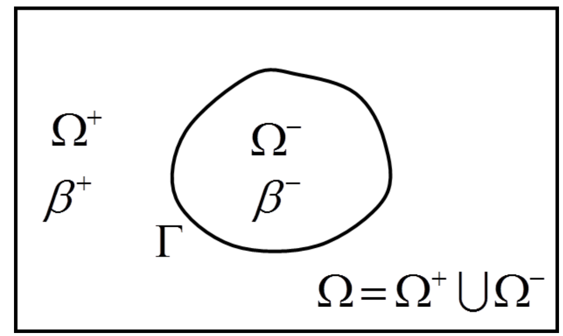
- Fast Fourier transform (FFT) O(N logN)
  - Augmented IIM *(Li 1998)*
  - Explicit Jump IIM *(Wiegmann, Bube, 2004)*
  - Augmented MIB *(Feng, Long, Zhao, 2019)*

- **All existing methods are second order accurate**

Goal: Fourth order + Fast Poisson solver

- Fourth order augmented MIB (AMIB4)

• BVP without interface (*Feng, Zhao, JCP, 109391, 2020*).



• BVP with interface (*Feng, Zhao, JCP, 109677, 2020*).

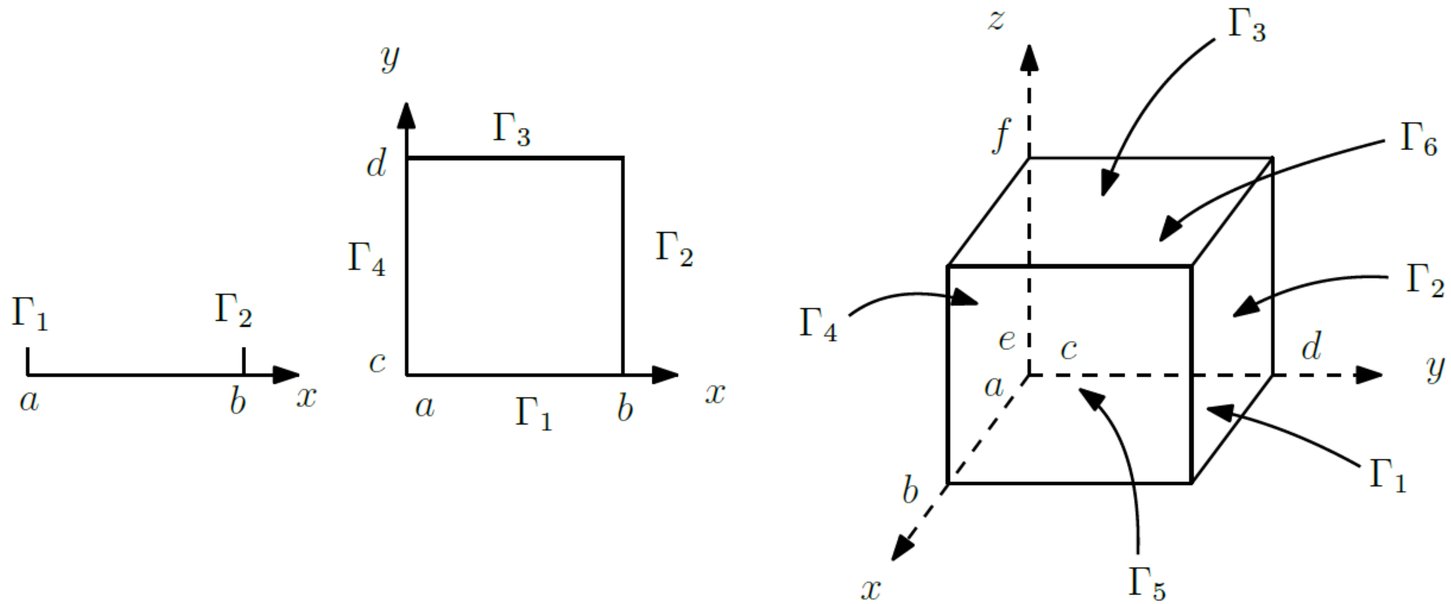
- Introduce auxiliary variables to preserve discrete Laplacian of central differences
- **FFT will not sense discontinuities**

2. Poisson boundary value problems

$$\Delta u = f(\vec{x}), \qquad (3)$$

subject to any boundary condition on Γ_j

1. Dirichlet: $u = \phi_j$
2. Neumann: $\dfrac{\partial u}{\partial n} = \phi_j$
3. Robin: $\dfrac{\partial u}{\partial n} + ku = \phi_j$



Fast Poisson Solvers

➢ **FISHPACK** *(Swarztrauber, Sweet)*

- The most widely used FFT-Poisson solver
- Second order central difference: $\{\frac{1}{h^2}, -\frac{2}{h^2}, \frac{1}{h^2}\}$
- Complexity in 3D: $O(n^3 \log n)$

$$u_{xx} = f, \qquad u(0) = u(1) = 0 \qquad (4)$$

1. Fast Sine transform (FST): $f \rightarrow \hat{f}$
2. $\hat{u} = \frac{\hat{f}}{\lambda}, \;\; \lambda_j = 4\, sin^2\left(\frac{j\pi}{n}\right)/(h^2)$
3. Inverse FST: $\hat{u} \rightarrow u$

FFT + fourth order central difference scheme

- Uniform grid: $\{x_0, x_1, \dots, x_n\}$
- Five points stencil: $\frac{1}{h^2}\{-\frac{1}{12}, \frac{4}{3}, -\frac{5}{2}, \frac{4}{3}, -\frac{1}{12}\}$

$$u_{xx} = f, \qquad u(0) = u(1) = 0 \qquad (4)$$

1. Fast Sine transform (FST): $f \to \hat{f}$
2. $\hat{u} = \frac{\hat{f}}{\lambda}, \;\; \lambda_j = -(\cos\left(\frac{j\pi}{n}\right) - 1)(\cos\left(\frac{j\pi}{n}\right) - 7)/(3h^2)$
3. Inverse FST: $\hat{u} \to u$

- Example 1

$u = \sin(4\pi x)$,
$f = 16\pi^2 \sin(4\pi x)$

- Example 2

$u = -\cos(4\pi x) + 1$,
$f = 16\pi^2 \cos(4\pi x)$

FFT vs LU decomposition

Example 1

N	FFT		LU	
	Error	Order	Error	Order
32	2.607E-4	–	2.607E-4	–
64	1.646E-5	3.99	1.646E-5	3.99
128	1.031E-6	4.00	1.031E-6	4.00
256	6.450E-8	4.00	6.450E-8	4.00
512	4.032E-9	4.00	4.032E-9	4.00

Example 2

N	FFT		LU	
	Error	Order	Error	Order
32	1.321E-2	–	5.229E-4	–
64	3.235E-3	2.03	3.294E-5	3.99
128	8.046E-4	2.00	2.063E-6	4.00
256	2.009E-4	2.00	1.290E-7	4.00
512	5.020E-5	2.00	8.061E-9	4.00

- **FFT degrades to second order in Example 2**

Anti-symmetric property

$$u(x_{-1}) = -u(x_1), \qquad u(x_{n+1}) = -u(x_{n-1}) \qquad (5)$$

- Example 1

$$u = \sin(4\pi x)\,,$$
$$f = 16\pi^2 \sin(4\pi x)$$

- Example 2

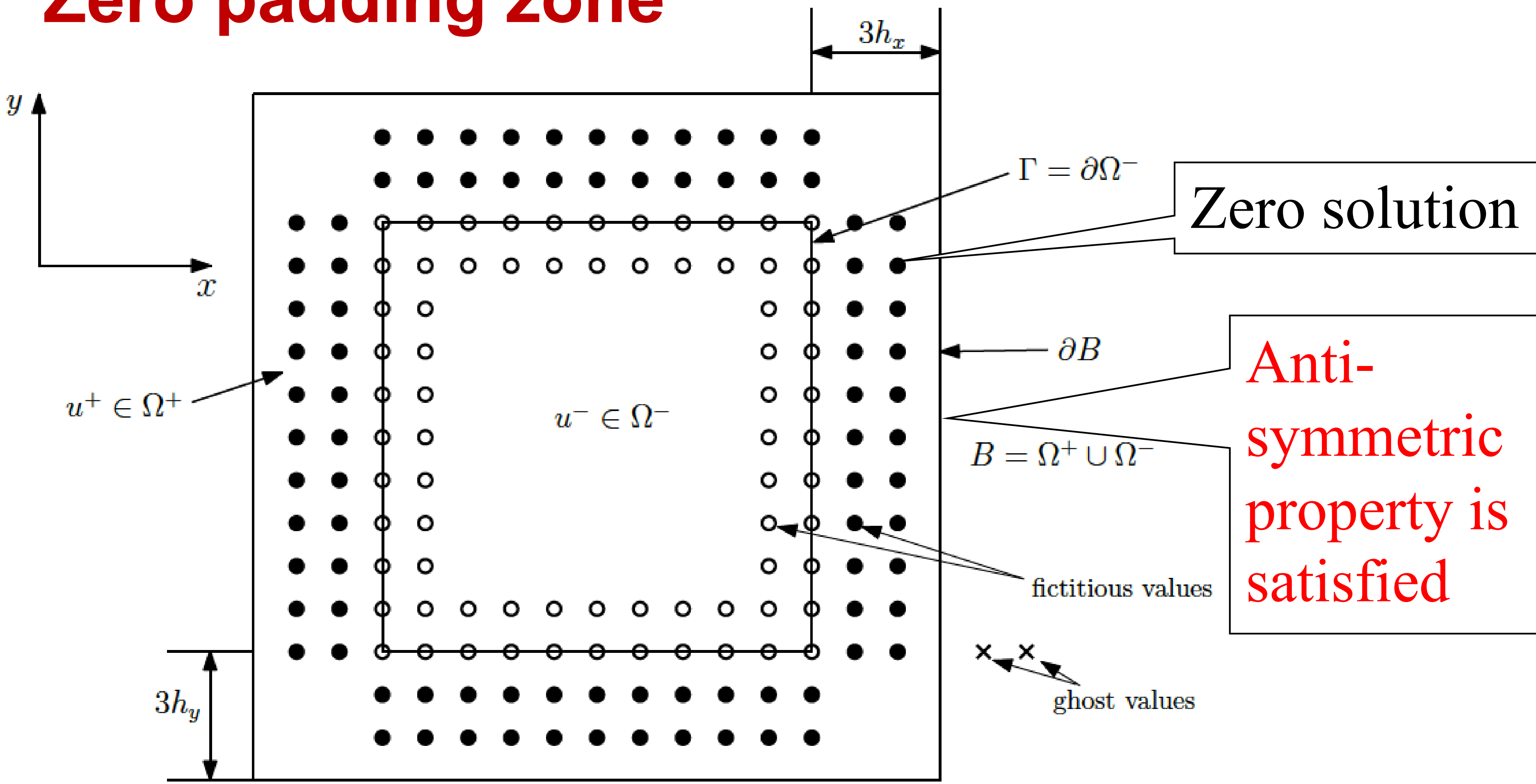
$$u = -\cos(4\pi x) + 1\,,$$
$$f = 16\pi^2 \cos(4\pi x)$$

- FFT degrades to second order, because (5) is not satisfied in Example 2

- **However, anti-symmetric property is invalid in general**

$$u_{xx} = f, \qquad u(0) = u(1) = 0 \qquad (4)$$

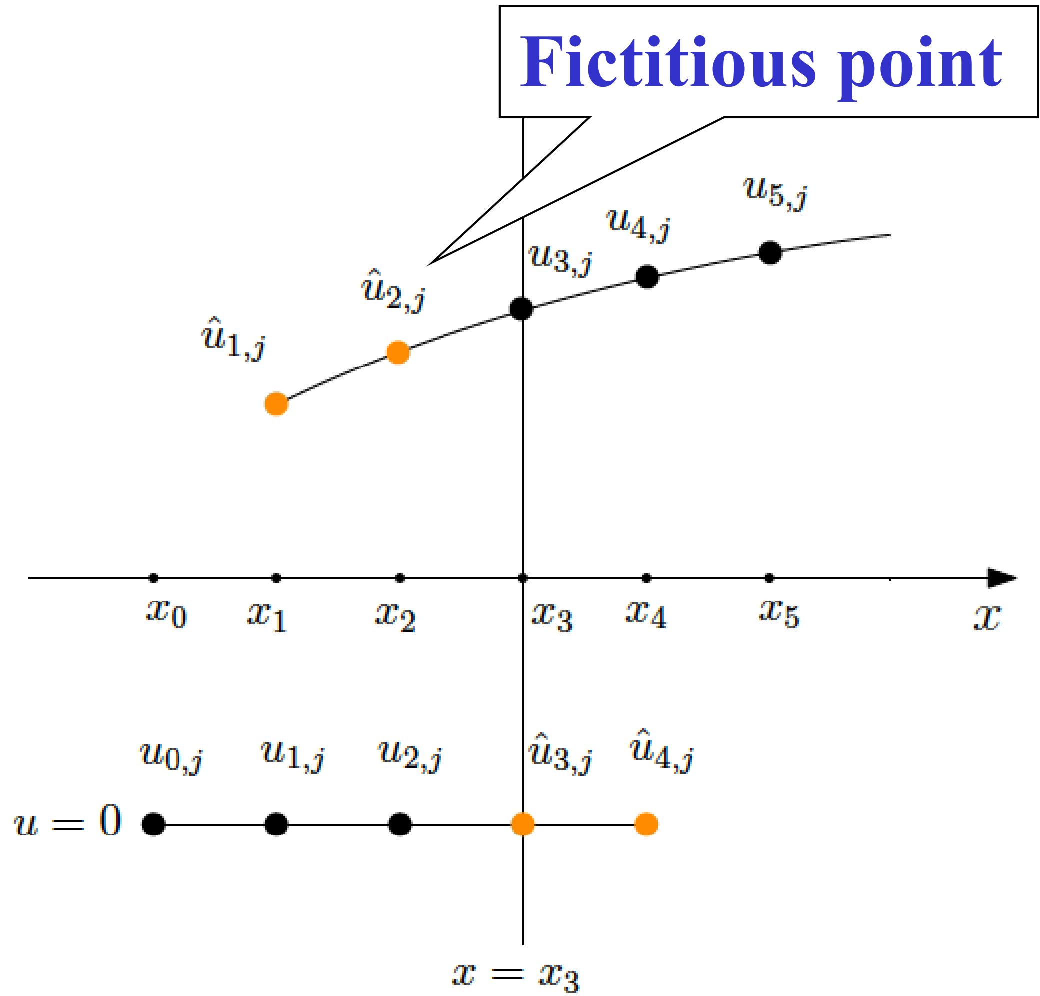
Zero padding zone



- Original boundary → immersed interface
- Boundary condition → interface condition

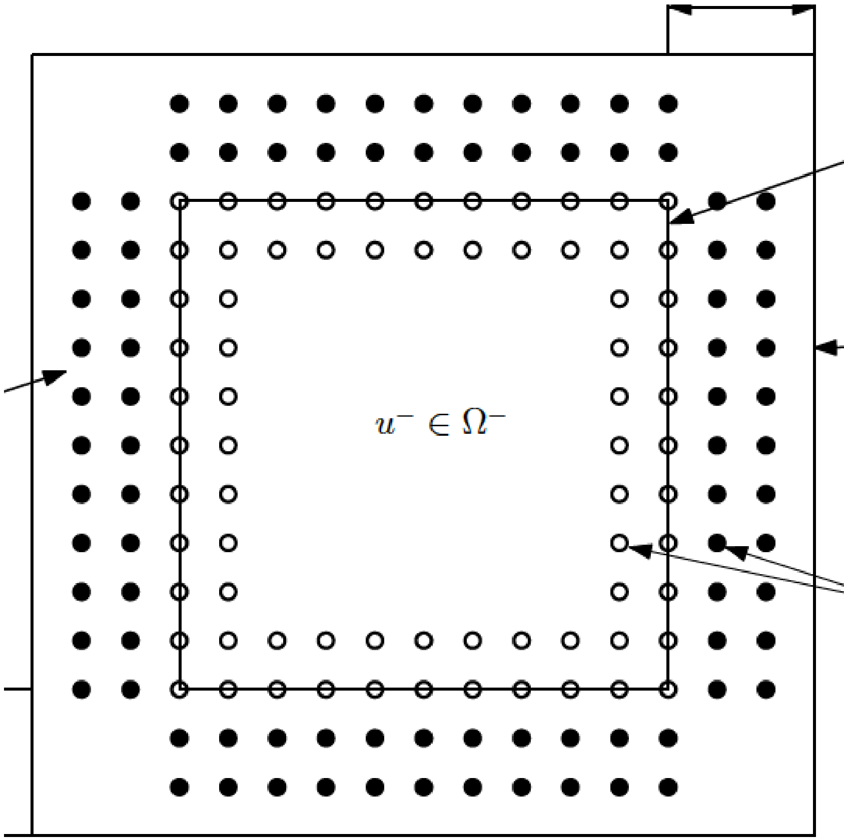
Matched interface and boundary (MIB) metho

- boundary condition: $$\frac{\partial u}{\partial x} + ku = \phi_3 \qquad (7)$$
- Second order MIB: Discretize (7) by using finite difference over $\{x_2, x_3, x_4\}$
- Solve fictitious value $\hat{u}_2$ as a linear combination of $\{u_3, u_4\}$ and ϕ_3
- High order: repeatedly enforce (7) to generate more fictitious values



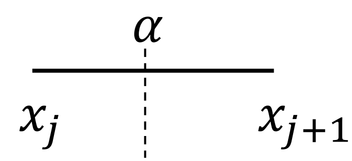
MIB fictitious value representation

- Two layers outside boundary for fourth order MIB
- A linear combination of some neighboring nodes and boundary values



$$\hat{u}_{i,j} = \sum_{(x_I, y_J) \in S_{i,j}} W_{I,J} u_{I,J} + W_0 \phi \,, \qquad (8)$$

Taylor expansion with jumps

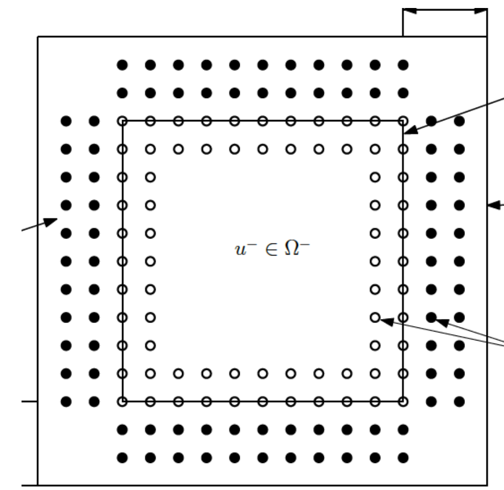


Theorem 1. Corrected fourth differences. Let $x_j \le \alpha < x_{j+1}, h^- = x_j - \alpha$, and $h^+ = x_{j+1} - \alpha$. Suppose $u \in C^6[x_j - 2h, \alpha) \bigcap C^6(\alpha, x_{j+1} + 2h]$, with derivative extending continuously up to the interface α. Then the following approximations hold to $O(h^4)$:

$$u_{xx}(x_{j-1}) \approx \frac{1}{h^2}[-\frac{1}{12}u(x_{j-3}) + \frac{4}{3}u(x_{j-2}) - \frac{5}{2}u(x_{j-1}) + \frac{4}{3}u(x_j) - \frac{1}{12}u(x_{j+1})] + \frac{1}{12h^2}\sum_{m=0}^{5}\frac{(h^+)^m}{m!}[u^{(m)}], \tag{22}$$

where $[u^{(m)}]$ are Cartesian derivative jumps

$$[\frac{\partial^m u}{\partial x^m}]|_{x=\alpha} = \lim_{x\to\alpha^+}\frac{\partial^m u}{\partial x^m} - \lim_{x\to\alpha^-}\frac{\partial^m u}{\partial x^m}$$



- Introduce Cartesian derivative jumps as auxiliary variables

$$Q := \{[u]_i, \left[\frac{\partial u}{\partial x}\right]_i, \left[\frac{\partial^2 u}{\partial x^2}\right]_i, \dots [u]_j, \left[\frac{\partial u}{\partial y}\right]_j, \left[\frac{\partial^2 u}{\partial y^2}\right]_j, \dots\}^T$$

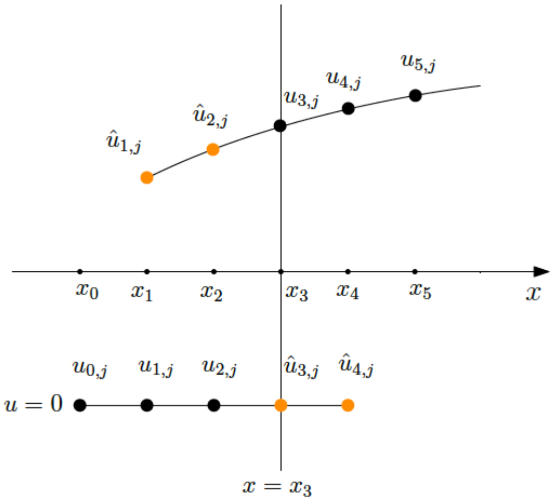
An augmented linear system

- Poisson equation (3) is discretized as

$$AU + BQ = F \qquad (9)$$

where *AU* is the discrete Laplacian generated by the 9-points fourth order central difference

- Reconstruct Cartesian derivative jumps by using fictitious values
- By eliminating fictitious values, Q linearly depends on U



$$CU + Q = \Phi \qquad (10)$$

A Schur complement procedure

$$\begin{aligned} AU + BQ &= F \\ CU + IQ &= \Phi \end{aligned} \qquad (11)$$

U: unknown solution; Q: auxiliary variable

- Solve a one-dimensionally smaller linear system iteratively

$$(\mathrm{I} - \mathrm{C}A^{-1}B)Q = \Phi - CA^{-1}F$$

- Then, solve U by one more FFT inversion

$$\mathrm{A}U = F - BQ$$

- FFT inversion complexity in 2D $\mathrm{O}(n^2 \log n)$

Fourth order MIB and AMIB

- 2D problem with Dirichlet boundaries
- AMIB=MIB+FFT, which is much faster than MIB

$[N_x, N_y]$	AMIB4					
	L_∞		L_2		CPU time(s)	iter no.
	Error	Order	Error	Order		
[11, 11]	4.408E-5	–	1.549E-5	–	4.823E-3	20
[27, 27]	1.096E-6	3.87	4.712E-7	3.65	1.290E-2	20
[59, 59]	4.662E-8	3.94	2.209E-8	3.81	4.056E-2	20
[123, 123]	2.442E-9	3.97	1.212E-9	3.90	0.139	21
[251, 251]	1.400E-10	3.98	7.111E-11	3.95	0.529	21
[507, 507]	7.771E-12	4.10	4.004E-12	4.08	2.403	21

$[N_x, N_y]$	MIB4				
	L_∞		L_2		CPU time(s)
	Error	Order	Error	Order	
[13, 13]	4.070E-5	–	1.841E-5	–	2.00E-3
[29, 29]	1.076E-6	4.28	5.267E-7	4.19	2.466E-2
[61, 61]	4.644E-8	4.12	2.345E-8	4.08	0.207
[125, 125]	2.444E-9	4.06	1.252E-9	4.04	1.836
[253, 253]	1.423E-10	4.00	7.329E-11	4.00	15.386
[509, 509]	3.363E-11	2.06	1.417E-11	2.34	148.41

Higher order AMIB

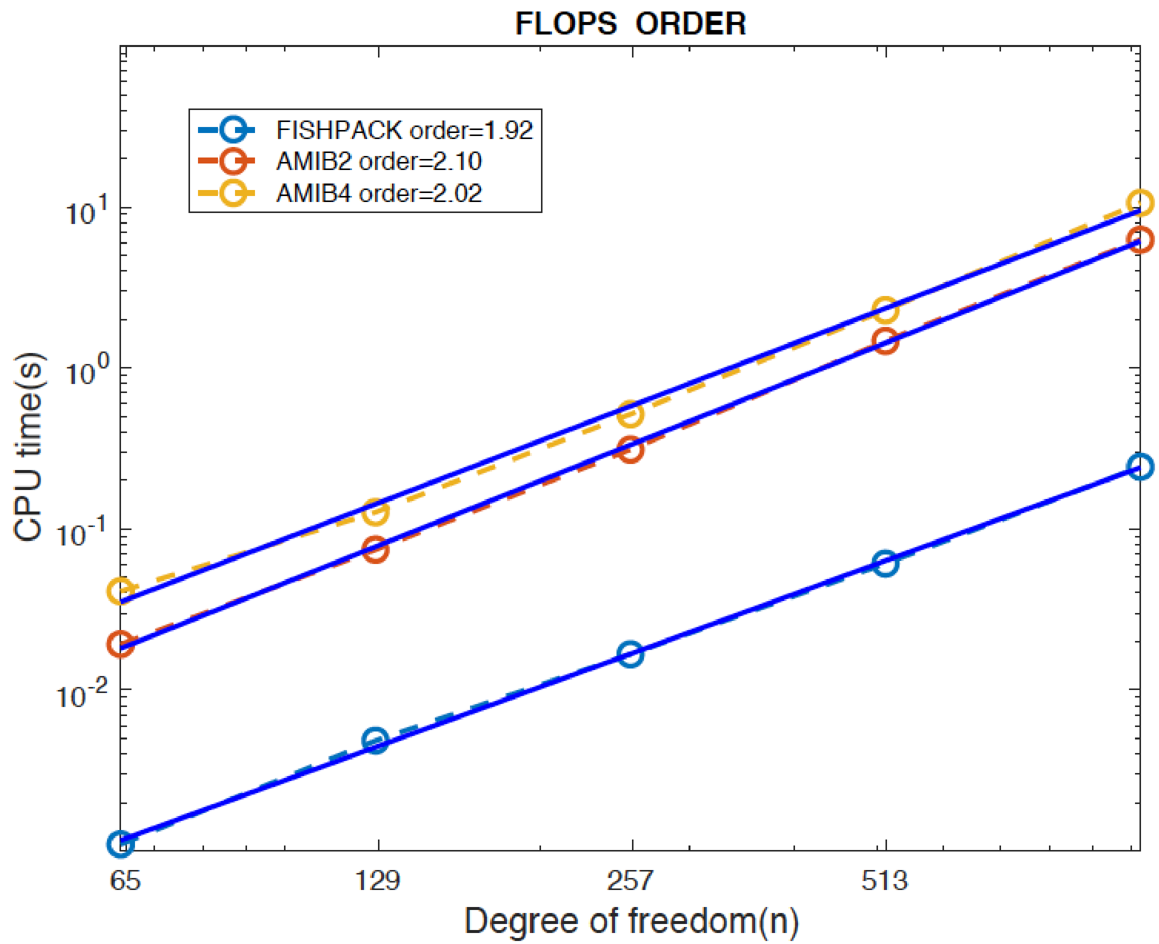
- Dirichlet, Robin, Robin, and Neumann

$[N_x, N_y]$	AMIB6 and $k = 10$				
	L_∞		L_2		
	Error	Order	Error	Order	
$[25, 25]$	1.072E-2	–	1.522E-3	–	32
$[57, 57]$	9.828E-5	5.54	1.640E-5	5.35	37
$[123, 123]$	9.412E-7	6.10	1.712E-7	5.98	40
$[249, 249]$	1.106E-8	6.12	2.119E-9	6.05	50
$[505, 505]$	1.489E-10	6.07	2.926E-11	6.04	69

$[N_x, N_y]$	AMIB8 and $k = 20$				
	L_∞		L_2		
	Error	Order	Error	Order	
$[23, 23]$	1.033	–	0.120	–	34
$[55, 55]$	2.589E-3	6.67	3.681E-4	6.44	42
$[121, 121]$	3.111E-6	8.60	2.575E-7	9.29	48
$[247, 247]$	1.724E-8	7.07	1.808E-9	6.74	56
$[503, 503]$	6.801E-11	7.76	8.209E-12	7.56	69

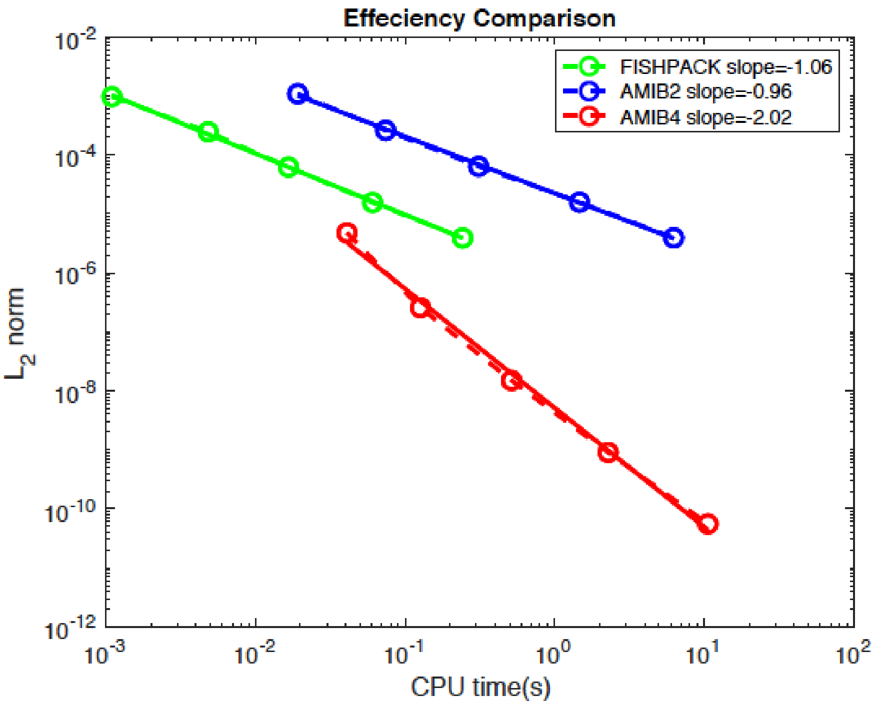
Efficiency in 2D

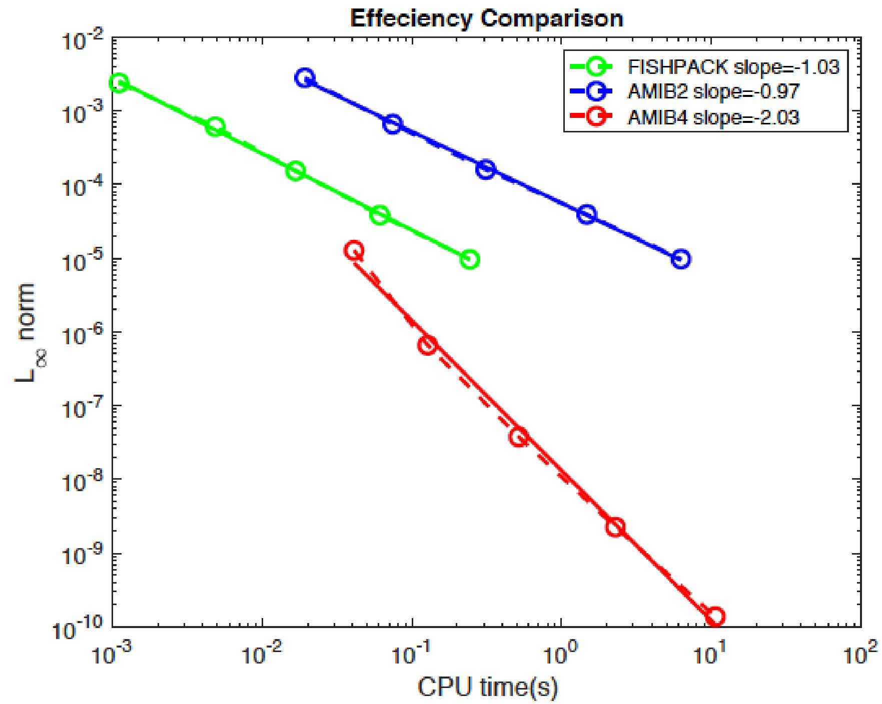
- FISHPACK *(Swarztrauber, Sweet)*
- AMIB: Performance is not optimized
- On the same mesh, AMIB2 and AMIB4 are more expensive than FISHPACK
- All: $O(n^2 \log n)$



Comparison with FISHPACK: cost-efficiency

- Plot error against CPU time





- **For a high precision, AMIB4 is more efficient that FISHPACK**
- Max error $= 10^{-10}$, FISHPACK is 10,000 times slower

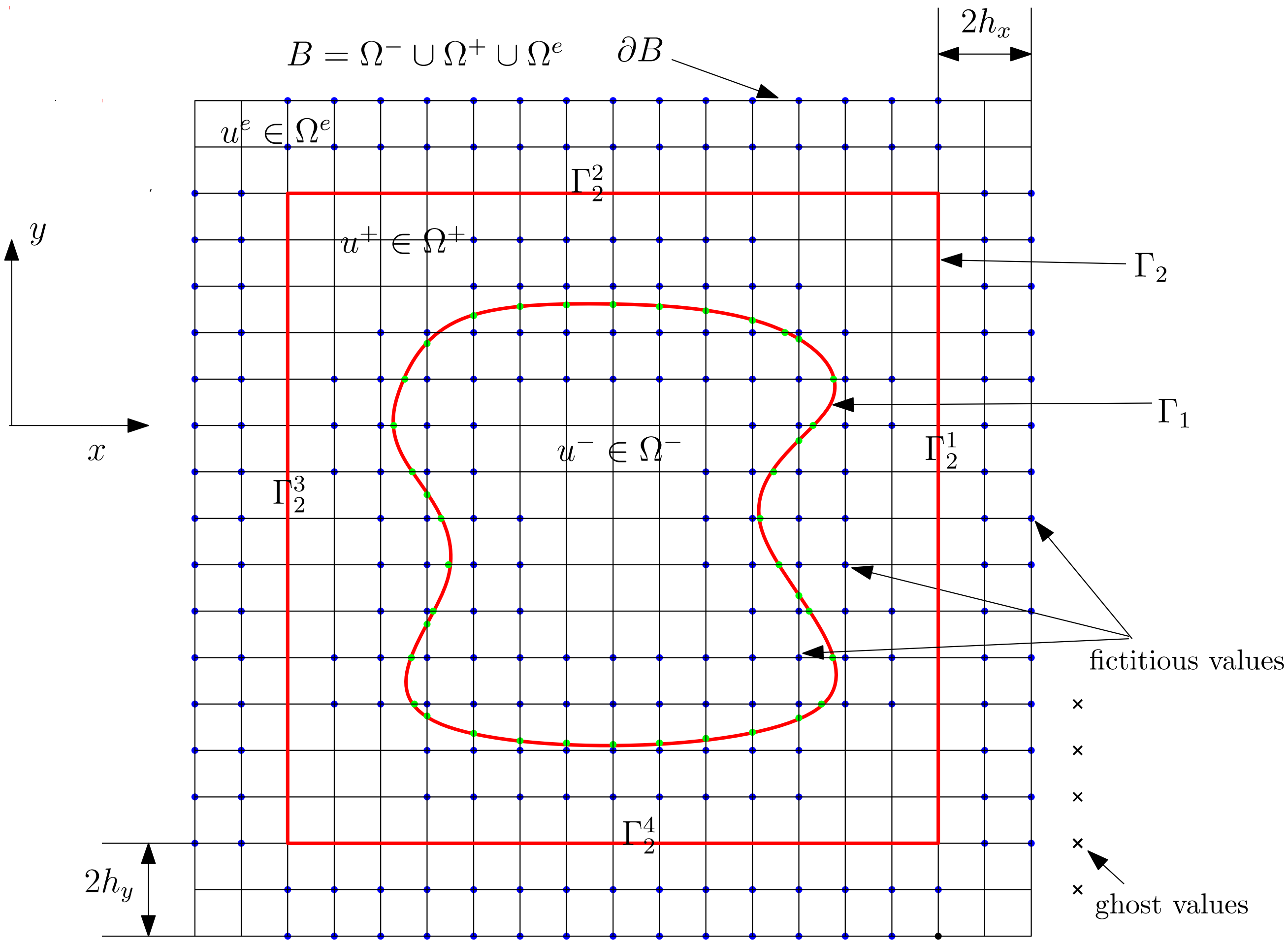
AMIB in 3D

- Dirichlet boundaries
- Tensor product type algorithm
- **Can be easily generalized to multi-dimensions**

$[N_x, N_y, N_z]$	AMIB4					
	L_∞		L_2		iter no.	CPU times(s)
	Error	Order	Error	Order		
[11, 11, 11]	0.870	–	0.127	–	37	4.722E-2
[27, 27, 27]	3.337E-2	3.81	3.347E-3	3.80	49	0.392
[59, 59, 59]	1.426E-3	4.13	1.216E-4	4.13	52	3.191
[123, 123, 123]	7.436E-5	4.10	5.760E-6	4.10	55	28.792
[251, 251, 251]	4.270E-6	4.06	3.132E-7	4.06	55	296.97

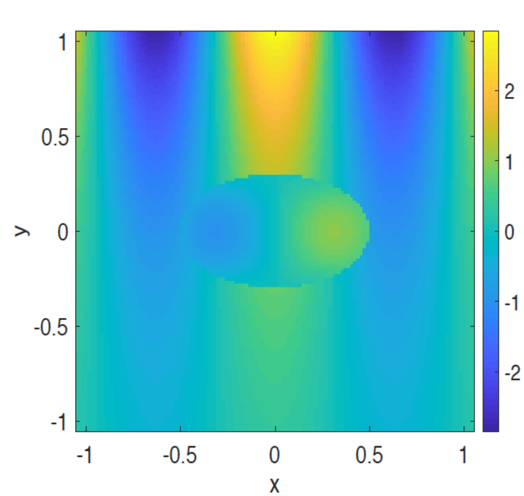
3. Elliptic interface problems in 2D

- Extend domain for FFT inversion
- MIB4 to handle interfaces and boundaries

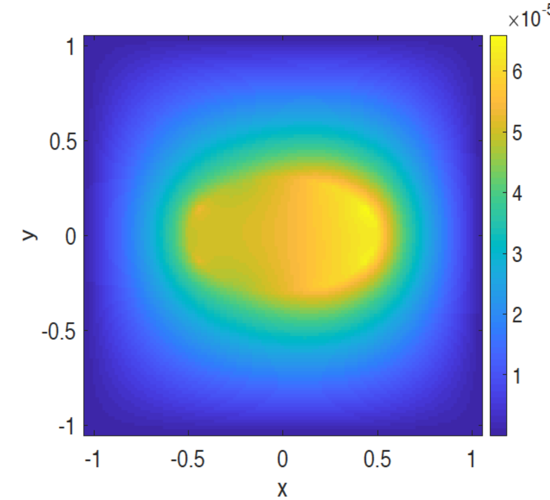


AMIB4 VS MIB4

- Fourth order
- AMIB is faster



Solution



Error

Table 5: Example 3b -$\beta^+ = 1, \beta^- = 10$; Ellipse interface; Robin boundary condition.

$[n, n]$	AMIB4					
	L_∞		L_2		iter no.	CPU time (s)
	Error	Order	Error	Order		
$[60, 60]$	1.647E-3	–	1.019E-3	–	39	3.967E-2
$[124, 124]$	1.132E-4	3.69	6.923E-5	3.70	62	0.171
$[252, 252]$	5.973E-6	4.15	3.559E-6	4.19	98	0.915
$[508, 508]$	4.264E-7	3.77	2.612E-7	3.73	141	5.387
$[n, n]$	MIB4					
	L_∞		L_2		iter no.	CPU time (s)
	Error	Order	Error	Order		
$[60, 60]$	1.190E-3	–	7.315E-4	–	509	0.115
$[124, 124]$	1.232E-4	3.12	7.559E-5	3.13	1286	0.803
$[252, 252]$	5.093E-6	4.49	3.000E-6	4.55	3366	8.47
$[508, 508]$	4.096E-7	3.60	2.506E-7	3.54	7569	79.53

Gradient recovery

- 4^{th} order in solution and gradient

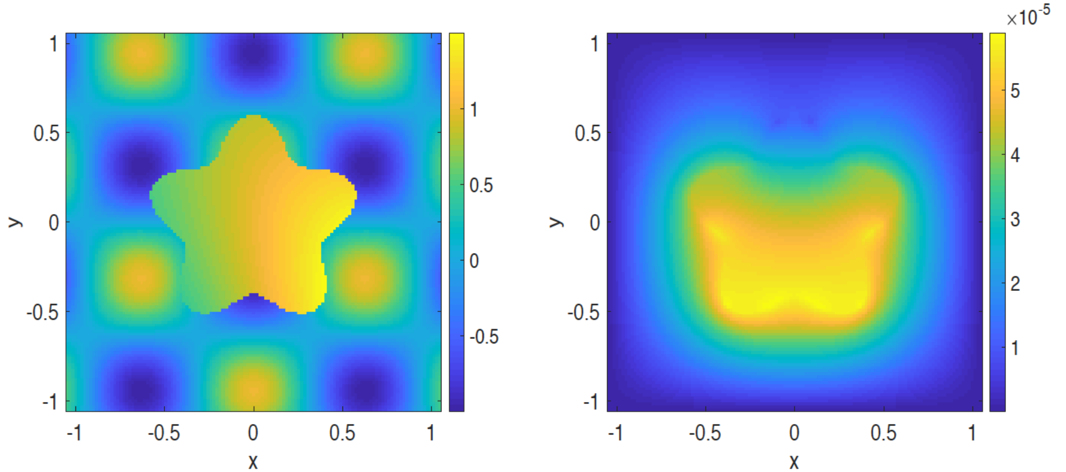
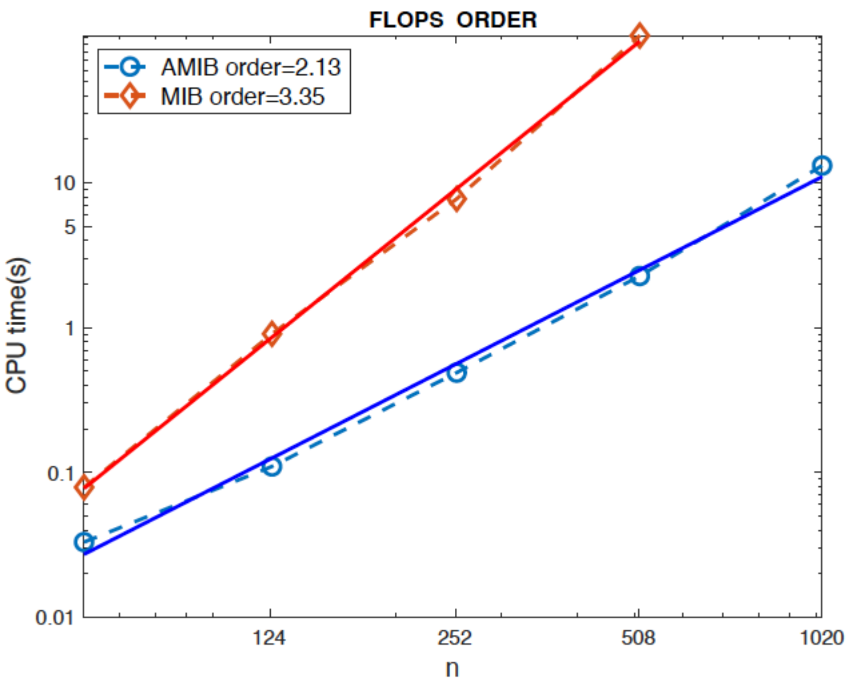


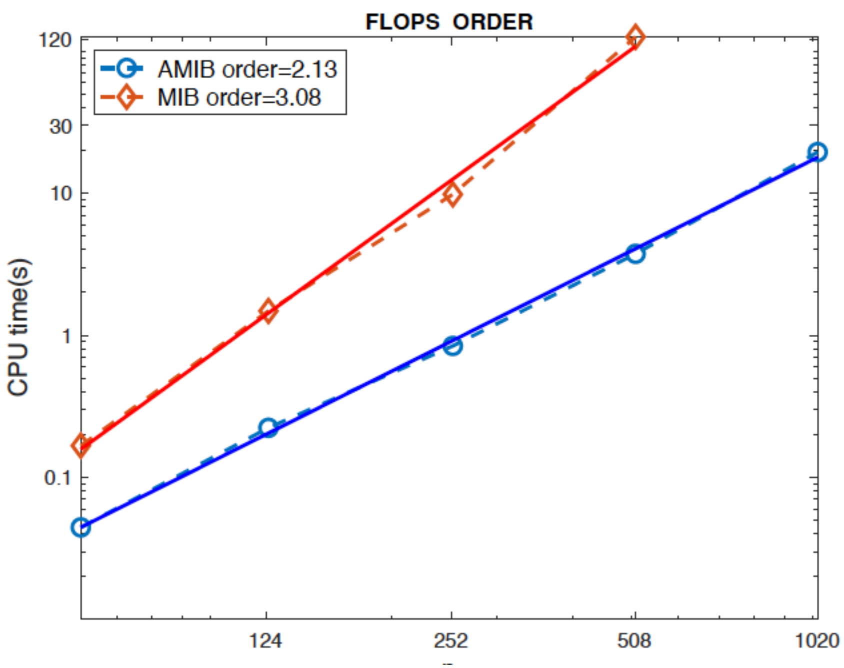
Table 7: Example 5 -$\beta^+ = 1, \beta^- = 20$; Five-leaf shaped interface.

$[n,n]$	AMIB4 L_∞ Error	L_∞ Order	L_2 Error	L_2 Order	iter no.	CPU time (s)
[60, 60]	1.078E-3	–	5.211E-4	–	43	4.442E-2
[124, 124]	5.885E-5	4.14	2.721E-5	4.13	70	0.223
[252, 252]	2.973E-6	4.06	1.229E-6	4.04	84	0.841
[508, 508]	9.272E-8	4.08	3.497E-8	4.08	100	3.744
[1020, 1020]	9.460E-9	2.76	2.263E-9	2.83	116	19.47
	Gradient L_∞ Error	L_∞ Order	L_2 Error	L_2 Order		
[60, 60]	4.824E-3	–	1.328E-3	–		
[124, 124]	2.300E-4	4.19	6.851E-5	4.08		
[252, 252]	3.221E-5	2.77	3.933E-6	4.03		
[508, 508]	8.286E-7	5.22	1.214E-7	4.96		
[1020, 1020]	3.586E-7	1.20	9.969E-9	3.59		

CPU time and efficiency for interface problem

- **MIB** $O(n^3)$
- **FFT-AMIB** $O(n^2)$ or $O(n^2 \log n)$





4. Summary

- Develop Augmented Matched Interface and Boundary (AMIB) method for solving elliptic PDEs
- **High order of accuracy**
  - 4^{th} – 8^{th} order accuracy for any boundary conditions
  - 4^{th} order accuracy for curve interfaces
- **High efficiency of the FFT**
  - $O(n^2 \log n)$ in 2D
  - $O(n^3 \log n)$ in 3D