Parallel
Discontinuous Galerkin Method

Yin Ki, NG
The Chinese University of Hong Kong

Aug 5, 2015

Mentors: Dr. Ohannes Karakashian, Dr. Kwai Wong

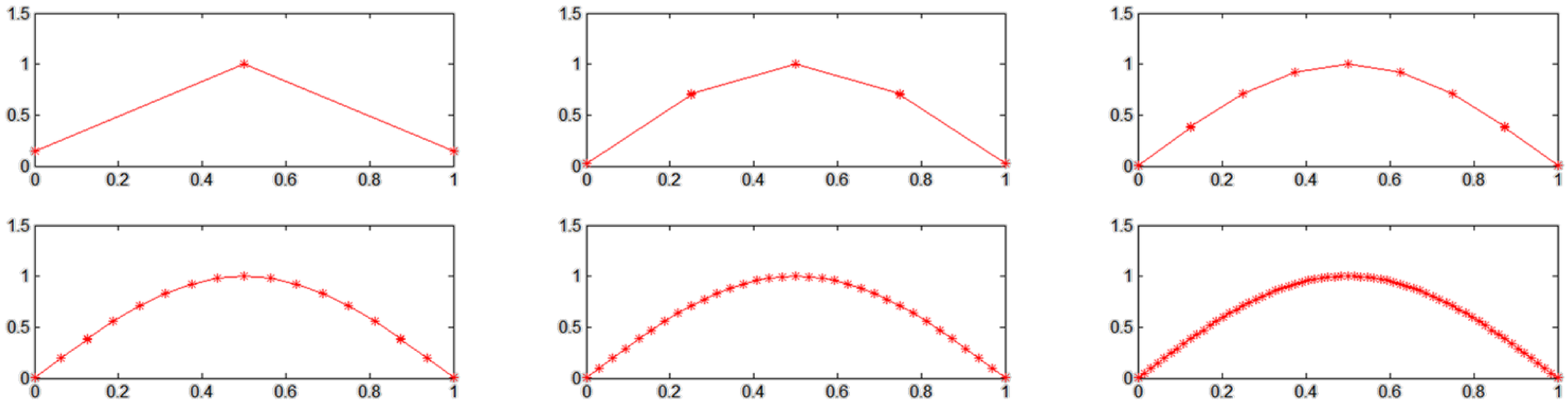
Overview

- Project Goal
  - Implement parallelization on Discontinuous Galerkin Method (DG-FEM)
  - scaled on existing supercomputers

- Overview
- Mathematics behind
- 1D Parallelization
- Future Work
- Acknowledgement

Discontinuous Galerkin Method (DG-FEM)

- Discontinuous Galerkin Method (DG-FEM)
  - A class of Finite Element Method (FEM)
  - Finding approximate solutions to system of differential equations



Example: Solving Heat Equation (1D)

- 1D Poisson's Equation on domain $I = [a, b]$

$$\begin{cases} -u'' = f \\ u(a) = u(b) = 0 \end{cases}$$

u: test function (to be solved)

Example: Solving Heat Equation (1D)

u: test function (to be solved)

- Multiply by an arbitrary trial function v

  (satisfying $v(a) = v(b) = 0$)

$$-u''v - fv = 0$$

- Integration (Strong-form)

$$-\int u''v - \int fv = 0$$

Example: 1D *Finite Element Method*

u: test function (to be solved)

v : trial function (our choice)

- Suppose v is continuous over I, integration by parts

$$\int u'v' + [u'v]_I - \int fv = 0$$

- Obtain Weak-form of *Finite Element Method* (FEM)

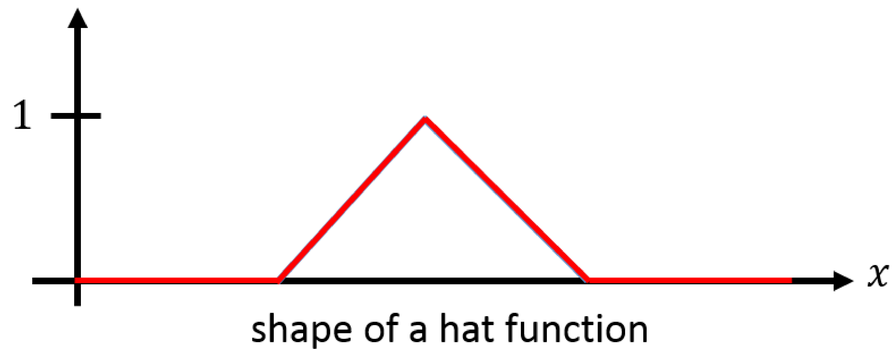
$$\int u'v' - \int fv = 0$$

Example: 1D *FEM* – Basis Functions

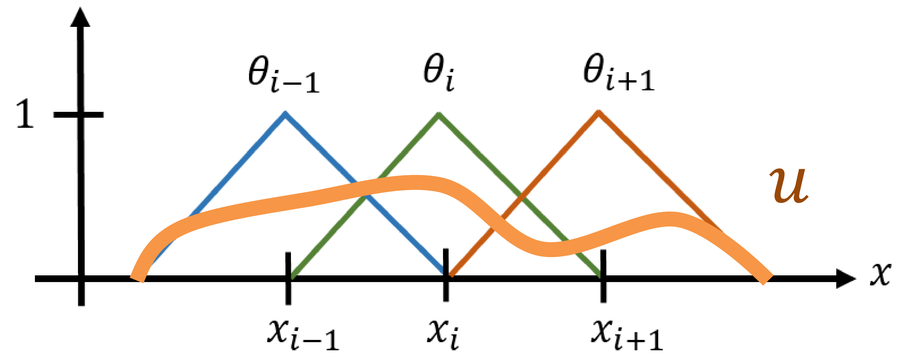
u: test function (to be solved) v : trial function (our choice)

(FEM)

- Choose continuous basis functions θ over I to approximate u
- Example: Hat function (Linear)



shape of a hat function



Example: 1D *FEM* – Basis Functions

u: test function (to be solved) v : trial function (our choice)

(FEM)

- Choose v to be the the basis functions θ
- Approximate $u_{\mathrm{i}} = \sum_j \hat{u}_j \, v_j$: linear combination of v_j

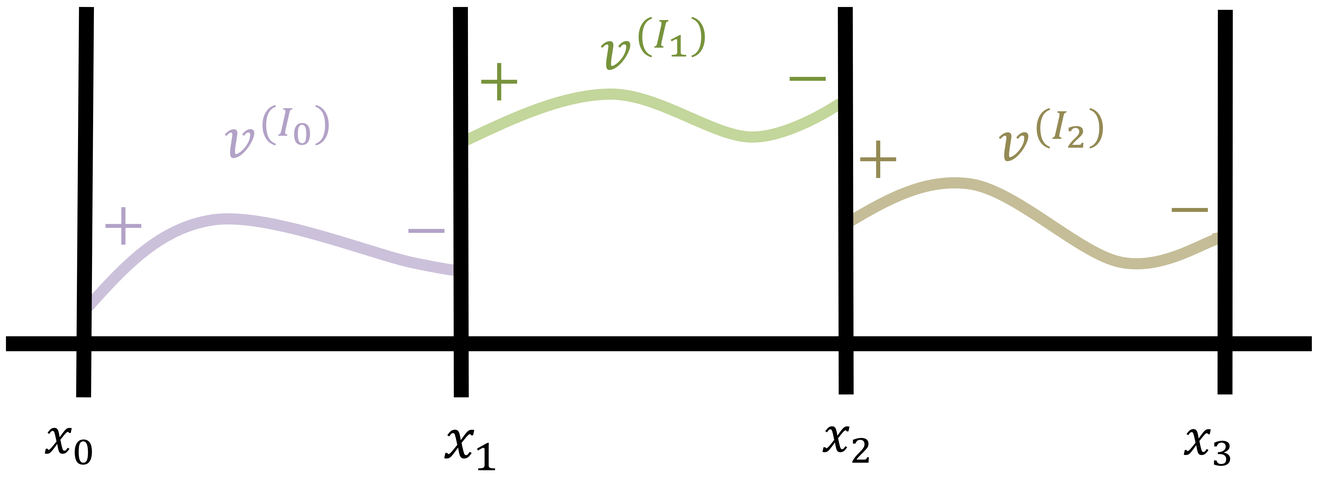
$$\int u'v' - \int fv = 0$$

$$\begin{bmatrix} \int_I v_0' v_0' & \int_I v_1' v_0' & \int_I v_2' v_0' \\ \int_I v_0' v_1' & \int_I v_1' v_1' & \int_I v_2' v_1' \\ \int_I v_0' v_2' & \int_I v_1' v_2' & \int_I v_2' v_2' \end{bmatrix} \begin{bmatrix} \hat{u}_0 \\ \hat{u}_1 \\ \hat{u}_2 \end{bmatrix} - \begin{bmatrix} \int_I f v_0 \\ \int_I f v_1 \\ \int_I f v_2 \end{bmatrix} = 0$$

Example: 1D *DG-FEM*

u: test function (to be solved) v : trial function (our choice)

- Suppose v is discontinuous on x_j



x_j^+: start of the “left” interval I_j

x_j^-: start of the “right” interval I_{j-1}

- Integration by parts on each interval

$$-\int u''v = -\sum_j \int_{I_j} u''\, v = \sum_j \left(\int_{I_j} u'\, v'\right) - \sum_j \left[u'v\right]_{x_j}^{x_{j+1}} \quad)$$

Example: 1D *DG-FEM*

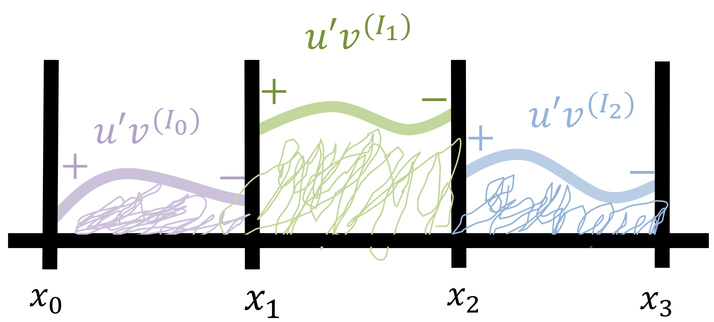
u: test function (to be solved)

v : trial function (our choice)

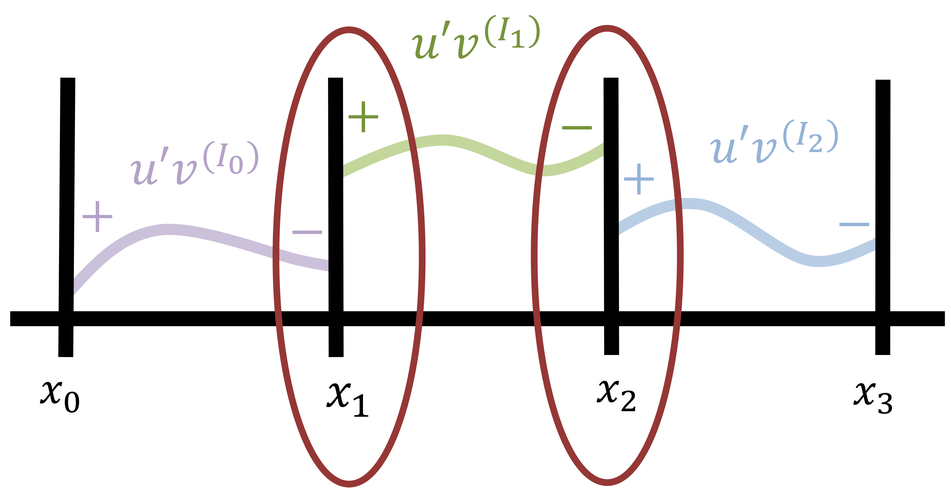
$$-\sum_j \left[u'\,v\right]_{x_j}^{x_{j+1}} = \sum_j u'(x_j^+)v(x_j^+) - u'(x_{j+1}^-)v(x_{j+1}^-)$$

$$= u'(x_0^+)v(x_0^+) - u'(x_1^-)v(x_1^-)$$
$$+ u'(x_1^+)v(x_1^+) - u'(x_2^-)v(x_2^-)$$
$$+ u'(x_2^+)v(x_2^+) - u'(x_3^-)v(x_3^-)$$

Jumps on internal nodes



$$= u'(x_0^+)v(x_0^+) + \sum_{\boldsymbol{interiorj}} \left(\boldsymbol{u'(x_j^+)v(x_j^+) - u'(x_j^-)v(x_j^-)}\right) - u'(x_3^-)v(x_3^-)$$

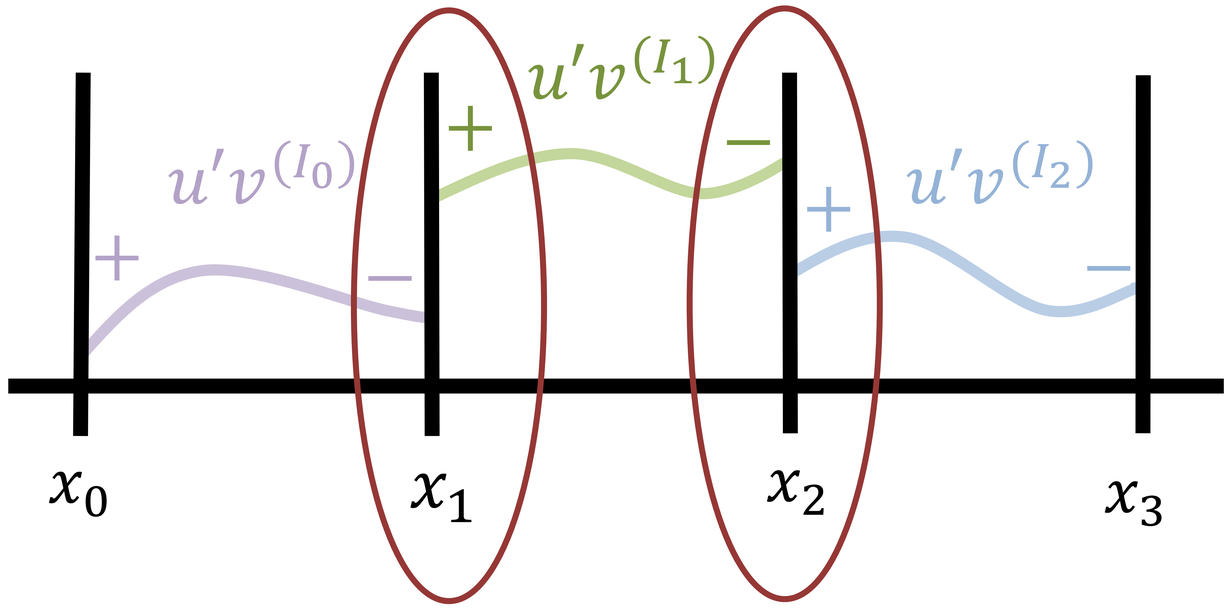


from interval-oriented to node-oriented

Example: 1D *DG-FEM*

u: test function (to be solved)

v : trial function (our choice)



x_j^+: start of the “left” interval I_j

x_j^-: start of the “right” interval I_{j-1}

- Non-zero jumps at interior nodes
- Transfer intervals information between intervals
- Discontinuous Galerkin Methods

Example: 1D *DG-FEM*

u: test function (to be solved)

v : trial function (our choice)

- Recall our Jumps and rewrite

$$[u'v]_{x_j} = \{u'\}_j[v]_j + \underbrace{[u']_j\{v\}_j}_{0}$$

$\{\cdot\}$: Average of jump

$[\cdot]$: Difference of jump

- Define Bilinear Function

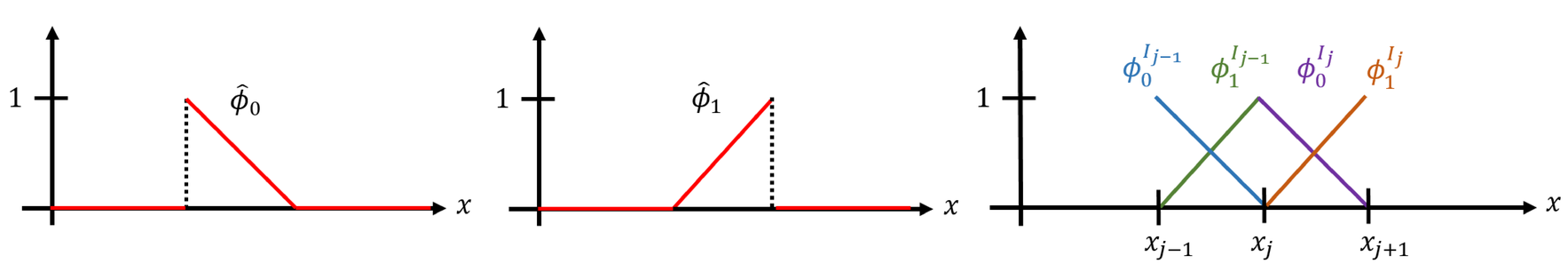
$$a(u,v) \equiv \sum_{j=0}^{j=N} \int_{x_j}^{x_{j+1}} u'v' + \sum_{j=0}^{j=N+1} \Big(\{u'\}_j[v]_j + \underbrace{\{v'\}_j[u]_j}_{\text{symmetric term}} \Big) + \underbrace{\gamma \sum_{j=0}^{N+1} \frac{1}{|I_j|}[u]_j[v]_j}_{\text{penalty term}}$$

- Construct Stiffness Matrix from $a(u,v)$

Example: 1D *DG-FEM* – Basis Functions

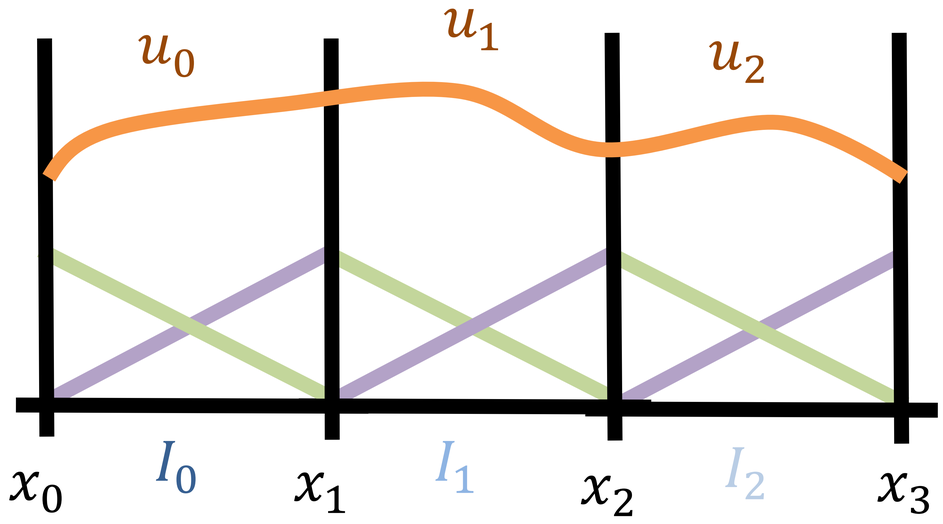
u: test function (to be solved) v : trial function (our choice)

- Choose basis function ϕ to approximate u
  - Example: Linear Lagrange polynomial



- $u_h = \sum_k \hat{u}_k^{(I_h)} \phi_k^{(I_h)} \in span(\phi)$

- $\hat{u}_k^{(I_h)}$: to be solved



Example: 1D *DG-FEM* – Basis Functions

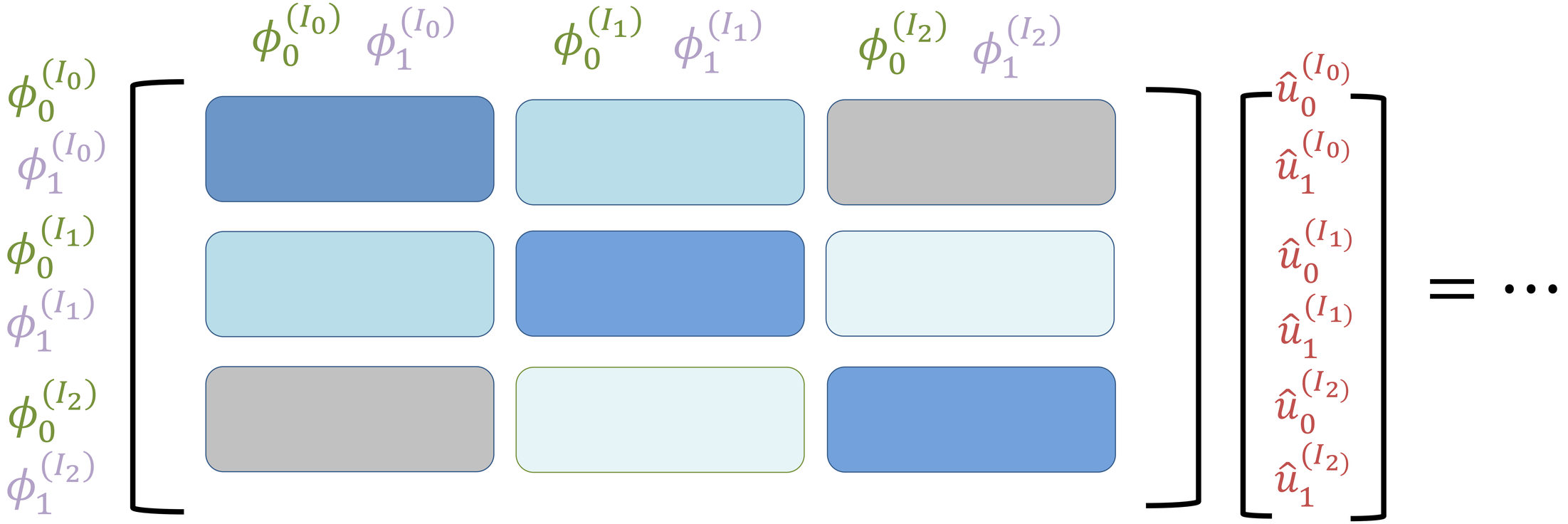
$u_h = \sum_k \hat{u}_k^{(I_h)} \phi_k^{(I_h)}$, $\hat{u}_k^{(I_h)}$ to be solved

$u \in span(\phi)$

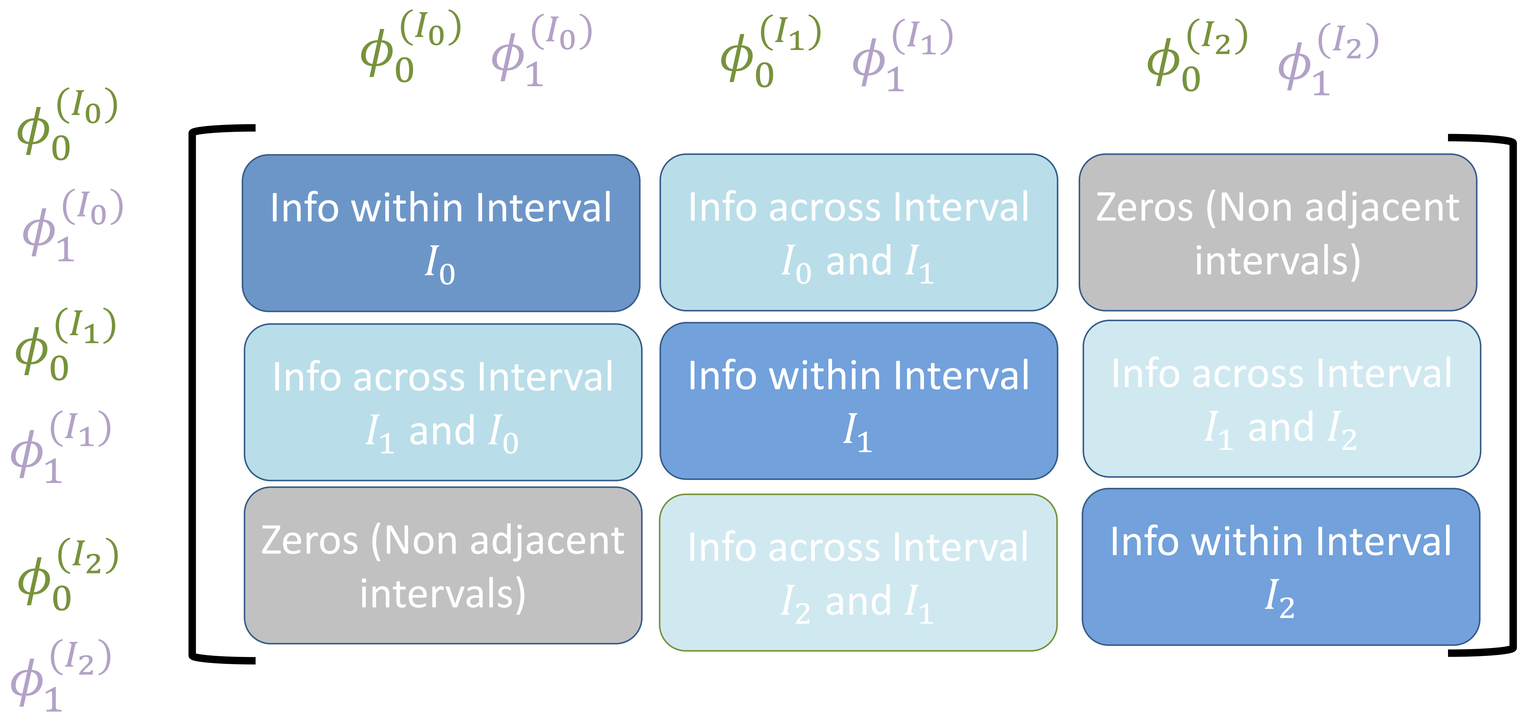
v: trial function (our choice)

$v = \phi$

- Also choose v to be ϕ
- Stiffness Matrix from $a(u, v)$=> from $a\left(\phi_{\{k_u\}}, \phi_{\{k_v\}}\right)$



Example: 1D *DG-FEM*



- Each block contains basis functions interaction w.r.t intervals
- Diagonal blocks: within an interval
- Sub-diagonal blocks: adjacent intervals
- Others: ZEROS (non-adjacent intervals)

Example: 1D *DG-FEM*

$u_h = \sum_k \hat{u}_k^{(I_h)} \phi_k^{(I_h)}$

$$u \in span(\phi)$$

v: trial function (our choice)

$$v = \phi$$

$$[u'v]_{x_j} = \{u'\}_j [v]_j + [u']_j \{v\}_j \quad (0)$$

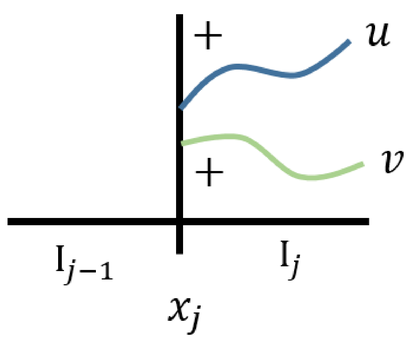
$$= \{\frac{1}{2}\left(u'(x_j^+) + u'(x_j^-)\right)\} [v(x_j^+) - v(x_j^-)]$$

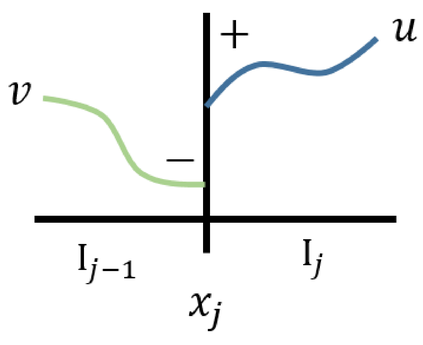
x_j^+: start of the "left" interval I_j

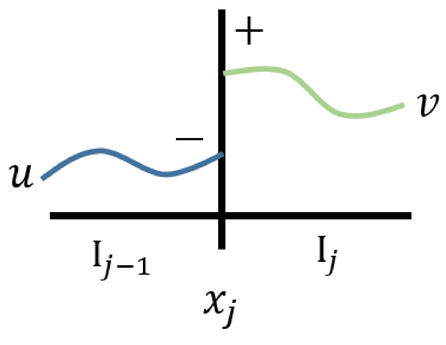
x_j^-: start of the "right" interval I_{j-1}

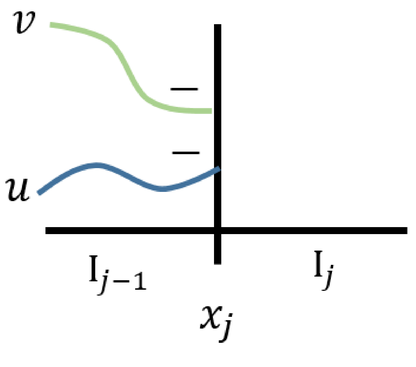
$$\frac{1}{2}u'(x_j^+)v(x_j^+) - \frac{1}{2}u(x_j^+)v(x_j^-) + \frac{1}{2}u'(x_j^-)v(x_j^+) - \frac{1}{2}u'(x_j^-)v(x_j^-)$$

"++" "+-" "-+" "--"









Example: 1D *DG-FEM*

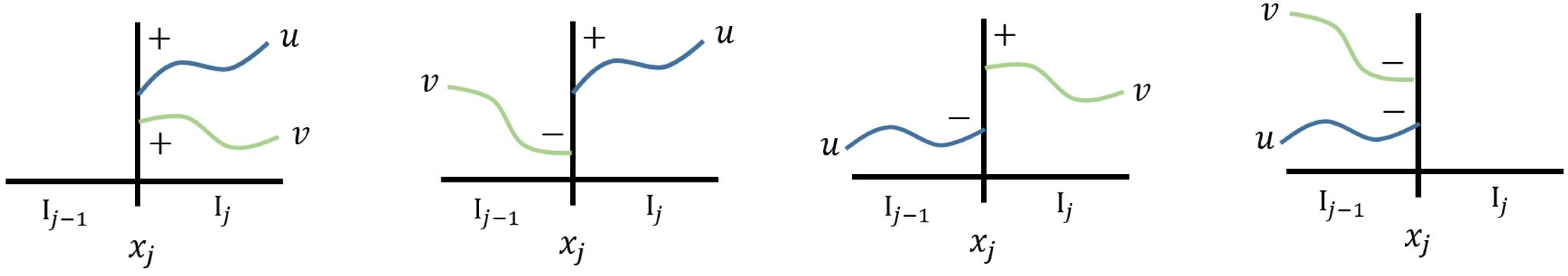
Example

- $j = 1$ (on x_1)

$\phi_0^{(I_0)}$ $\phi_1^{(I_0)}$	“--”	“+-”	Zeros (Non adjacent intervals)
$\phi_0^{(I_1)}$ $\phi_1^{(I_1)}$	“-+”	“++”	Info across Interval I_1 and I_2
$\phi_0^{(I_2)}$ $\phi_1^{(I_2)}$	Zeros (Non adjacent intervals)	Info across Interval I_2 and I_1	Info within Interval I_2

$$\frac{1}{2}u'(x_j^+)v(x_j^+) - \frac{1}{2}u(x_j^+)v(x_j^-) + \frac{1}{2}u'(x_j^-)v(x_j^+) - \frac{1}{2}u'(x_j^-)v(x_j^-)$$

“++” “+-” “-+” “--”



Example: 1D *DG-FEM* – Correctness

u: test function (to be solved)

v : trial function (our choice)

$\{\cdot\}$: Average of jump

$[\cdot]$: Difference of jump

- Recall Original Equation

$$-\int u''v = -\sum_j \int_{I_j} u''\,v =$$

$$\sum_j \left(\int_{I_j} u'\,v'\right) - \sum_j \left[u'v\right]_{x_j}^{x_{j+1}} \quad)$$

- Bilinear Function: terms added

$$a(u,v) \equiv \sum_{j=0}^{j=N} \int_{x_j}^{x_{j+1}} u'v' + \sum_{j=0}^{j=N+1} \left(\{u'\}_j [v]_j + \underbrace{\{v'\}_j [u]_j}_{\text{symmetric term}} \right) + \underbrace{\gamma \sum_{j=0}^{N+1} \frac{1}{|I_j|} [u]_j [v]_j}_{\text{penalty term}}$$

Example: 1D *DG-FEM* – Correctness

u: test function (to be solved) v : trial function (our choice)

$$a(u,v) \equiv \sum_{j=0}^{j=N} \int_{x_j}^{x_{j+1}} u'v' + \sum_{j=0}^{j=N+1} \Big(\{u'\}_j [v]_j + \underbrace{\{v'\}_j [u]_j}_{\text{symmetric term}} \Big) + \underbrace{\gamma \sum_{j=0}^{N+1} \frac{1}{|I_j|} [u]_j [v]_j}_{\text{penalty term}}$$

- Symmetric term
  - Added for the matrix to be symmetric

- Penalty term
  - Added for the matrix to be positive-definite

Example: 1D *DG-FEM* – Correctness

u: test function (to be solved) v : trial function (our choice)

$$a(u,v) \equiv \sum_{j=0}^{j=N} \int_{x_j}^{x_{j+1}} u'v' + \sum_{j=0}^{j=N+1} \Big(\{u'\}_j [v]_j + \underbrace{\{v'\}_j [u]_j}_{\text{symmetric term}} \Big) + \underbrace{\gamma \sum_{j=0}^{N+1} \frac{1}{|I_j|} [u]_j [v]_j}_{\text{penalty term}}$$

- Show stiffness matrix postive-definite
  - Show

$$a(u,v) \geq A \, {u_{1,h}}^2 \quad \forall u \in V_h$$

- V_h: DG Finite Element Space $u_{1,h}$: well-defined norm

Example: 1D *DG-FEM* – Correctness

u: test function (to be solved) v : trial function (our choice)

Show $a(u,v) \geq A\, {u_{1,h}}^2 \quad \forall u \in V_h$

- V_h: DG Finite Element Space $u_{1,h}$: well-defined norm

- 1. Define $u_{1,h} = \left(\sum_{I_j} {u'_{I_j}}^2 + \sum_{I_j} {\{u'\}_j}^2 I_j + \sum_{I_j} \frac{\gamma}{I_j} {[u]_j}^2 \right)^{\frac{1}{2}}$

  - can be shown well-defined.

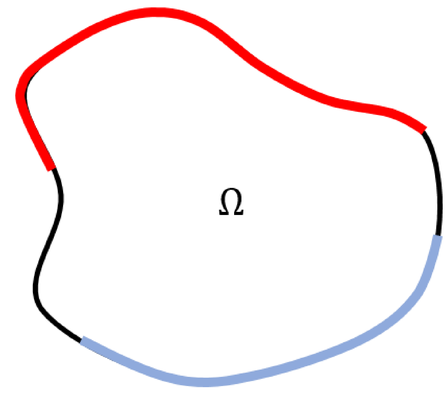
Example: 1D *DG-FEM* – Correctness

u: test function (to be solved) v : trial function (our choice)

$$\text{Show} \quad a(u,v) \geq A\, {u_{1,h}}^2 \quad \forall u \in V_h$$

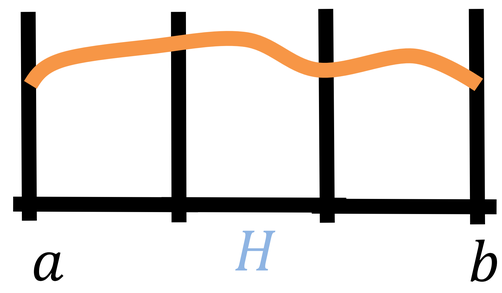
- 2. Important Inequalities used
  - Trace Inequality

$$\int_{\partial\Omega} v^2 ds \;\leq\; cH^{-1}\, {v_\Omega}^2 + cH\, {\nabla_\Omega}^2$$



  - Trace Inequality (1D)

$$v(a)^2 + v(b)^2 \;\leq\; cH^{-1}\, {v_I}^2 + cH\, {v'_I}^2$$



    - H: Length of Interval

Example: 1D *DG-FEM* – Correctness

u: test function (to be solved) v : trial function (our choice)

$$\text{Show} \quad a(u,v) \geq A\, {u_{1,h}}^2 \quad \forall u \in V_h$$

- 2. Important Inequalities used
  - Inverse Inequality

$$u'^2 \;\leq\; c\, I^{-2} {u_I}^2$$

  - Trace + Inverse: can show that

$$\sum_{I_j} \{u'\}_j \, [u]_j \;\leq\; c_1\, \delta \sum_{I_j} {u'_{I_j}}^2 + \frac{c_2}{\delta} \sum_{I_j} \frac{1}{I_j} \, {[u]_j}^2$$

Example: 1D *DG-FEM* – Correctness

u: test function (to be solved) v : trial function (our choice)

$$\text{Show} \quad a(u,v) \geq A\, {u_{1,h}}^2 \quad \forall u \in V_h$$

- After proof of $\sum_{I_j} \{u'\}_j [u]_j \leq c_1 \delta \sum_{I_j} {u'_{I_j}}^2 + \frac{c_2}{\delta} \sum_{I_j} \frac{1}{I_j} {[u]_j}^2$

$$a(u,u) - A\, {u_{1,h}}^2$$

$$= \left(\sum_{I_j} {u'_{I_j}}^2 - 2 \sum_{I_j} \{u'\}_j [u']_j + \sum_{I_j} \frac{\gamma}{I_j} {[u]_j}^2 \right)$$

$$- A \left(\sum_{I_j} {u'_{I_j}}^2 + \sum_{I_j} I_j \{u'\}_j{}^2 + \sum_{I_j} \frac{\gamma}{I_j} {[u]_j}^2 \right) \geq (1-A) \sum_{I_j} {u'_{I_j}}^2 + (1-A) \sum_{I_j} \frac{\gamma}{I_j} {[u]_j}^2$$

$$- \left(c_1 \delta \sum_{I_j} {u'_{I_j}}^2 + \frac{c_2}{\delta} \sum \frac{1}{I_j} {u_{I_j}}^2 \right) - c_3 A \sum_{I_j} {u'_{I_j}}^2$$

$$= (1 - A - c_1\delta - c_3 A) \sum_{I_j} {u'_{I_j}}^2 + (\gamma - \gamma A - \frac{c_2}{\delta}) \sum_{I_j} \frac{1}{I_j} {[u]_j}^2$$

$$\geq 0$$

Example: 1D *DG-FEM* – Correctness

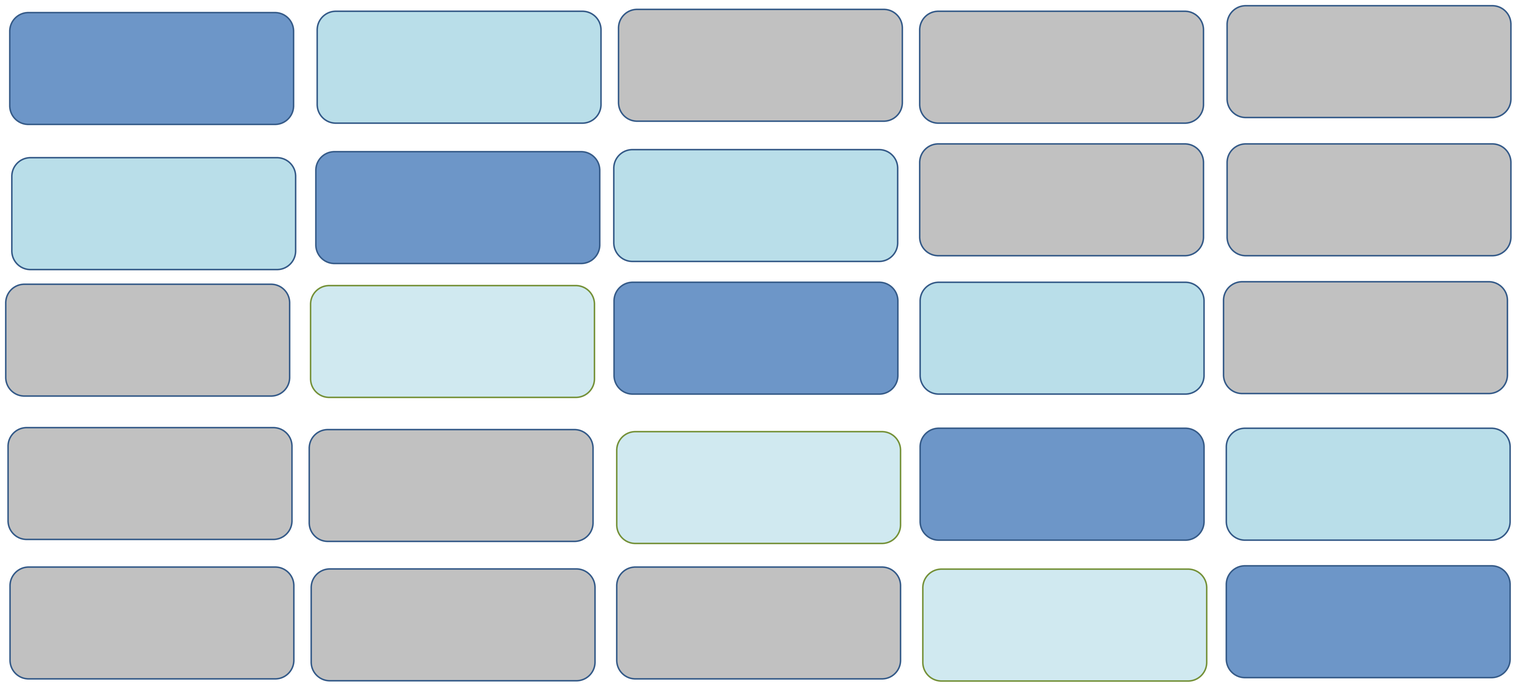
u: test function (to be solved) v : trial function (our choice)

$$a(u,v) \equiv \sum_{j=0}^{j=N} \int_{x_j}^{x_{j+1}} u'v' + \sum_{j=0}^{j=N+1} \Big(\{u'\}_j [v]_j + \underbrace{\{v'\}_j [u]_j}_{\text{symmetric term}} \Big) + \underbrace{\gamma \sum_{j=0}^{N+1} \frac{1}{|I_j|} [u]_j [v]_j}_{\text{penalty term}}$$

- Choice of γ
  - As long as it is large enough
  - Dependent on degree of basis functions

Example: 1D *DG-FEM*

- Tri-diagonal Matrix
- Symmetric
- Positive Definite
- Sparse

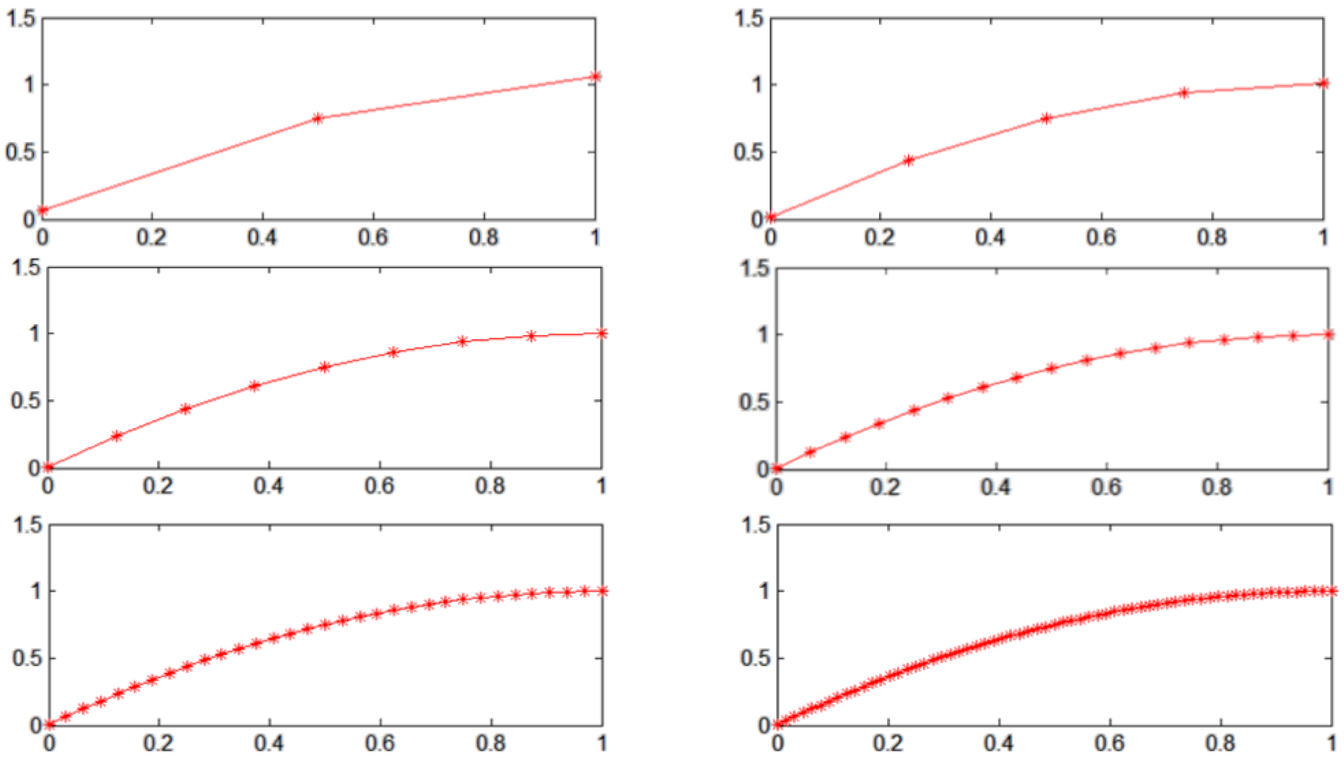


- Overview
- Mathematics behind
- 1D Parallelization
- Future Work
- Acknowledgement

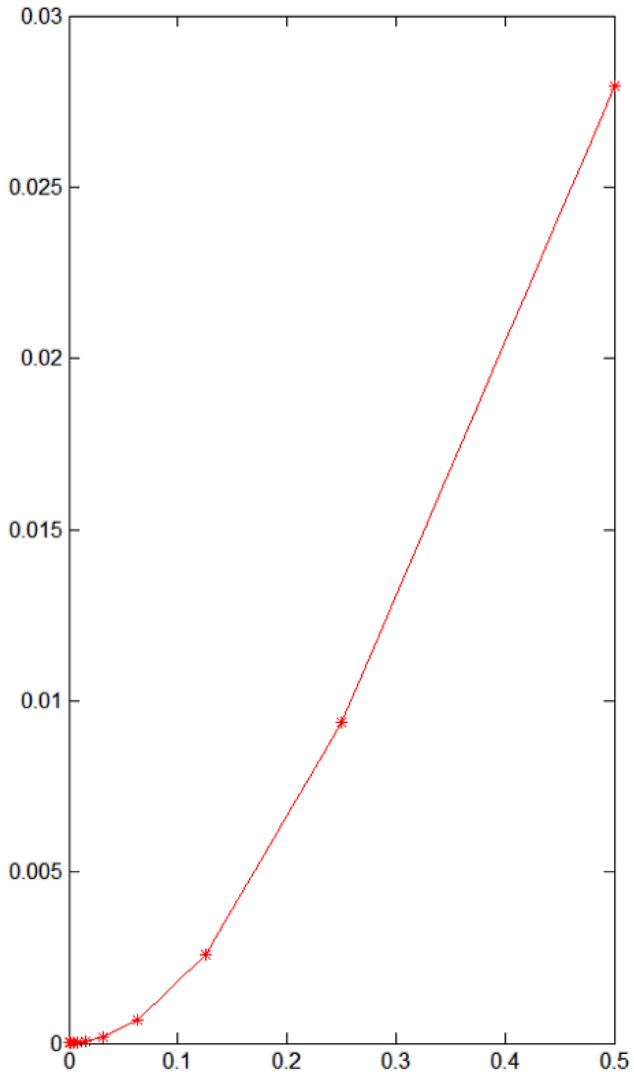
1D Parallelization: Why

- Higher accuracy …

Approximation of u_h using number of cells = 2, 4, 8, 16, 32, 64



Error norm of u_h against interval size



…leads to larger matrix (and cost)

1D Parallelization(1): Matrix Construction

- Observe: each interval/node contributes to a local matrix

$$a(u,v) \equiv \sum_{j=0}^{j=N} \int_{x_j}^{x_{j+1}} u'v' + \sum_{j=0}^{j=N+1} \Big(\{u'\}_j [v]_j + \underbrace{\{v'\}_j [u]_j}_{\text{symmetric term}} \Big) + \underbrace{\gamma \sum_{j=0}^{N+1} \frac{1}{|I_j|} [u]_j [v]_j}_{\text{penalty term}}$$

Step 3.1. $\int_{I_j} u'v'$

Local matrices

$$\begin{pmatrix} 4.00 & -4.00 \\ -4.00 & 4.00 \end{pmatrix}$$

$\phi_0^{(I_0)}$	$\phi_1^{(I_0)}$	$\phi_0^{(I_1)}$	$\phi_1^{(I_1)}$	$\phi_0^{(I_2)}$	$\phi_1^{(I_2)}$	$\phi_0^{(I_2)}$	$\phi_1^{(I_2)}$
4.00	−4.00	0.00	0.00	0.00	0.00	0.00	0.00
−4.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	4.00	−4.00	0.00	0.00	0.00	0.00
0.00	0.00	−4.00	4.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	4.00	−4.00	0.00	0.00
0.00	0.00	0.00	0.00	−4.00	4.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	4.00	−4.00
0.00	0.00	0.00	0.00	0.00	0.00	−4.00	4.00

1D Parallelization(1): Matrix Construction

- Observe: each interval/node contributes to a local matrix

$$a(u,v) \equiv \sum_{j=0}^{j=N} \int_{x_j}^{x_{j+1}} u'v' + \sum_{j=0}^{j=N+1} \Big(\{u'\}_j [v]_j + \underbrace{\{v'\}_j [u]_j}_{\text{symmetric term}} \Big) + \underbrace{\gamma \sum_{j=0}^{N+1} \frac{1}{|I_j|} [u]_j [v]_j}_{\text{penalty term}}$$

Step 3.2. $\{u'\}[v] + \{v'\}[u]$

Local matrices

$$\begin{pmatrix} -8.00 & 4.00 \\ 4.00 & 0.00 \end{pmatrix}$$

$$\begin{bmatrix} 0.00 & 2.00 & -2.00 & 0.00 \\ 2.00 & -4.00 & 4.00 & -2.00 \\ -2.00 & 4.00 & -4.00 & 2.00 \\ 0.00 & -2.00 & 2.00 & 0.00 \end{bmatrix}$$

$$\begin{pmatrix} 0.00 & 4.00 \\ 4.00 & -8.00 \end{pmatrix}$$

$\phi_0^{(I_0)}$	$\phi_1^{(I_0)}$	$\phi_0^{(I_1)}$	$\phi_1^{(I_1)}$	$\phi_0^{(I_2)}$	$\phi_1^{(I_2)}$	$\phi_0^{(I_2)}$	$\phi_1^{(I_2)}$
−8.00	6.00	−2.00	0.00	0.00	0.00	0.00	0.00
6.00	−4.00	4.00	−2.00	0.00	0.00	0.00	0.00
−2.00	4.00	−4.00	4.00	−2.00	0.00	0.00	0.00
0.00	−2.00	4.00	−4.00	4.00	−2.00	0.00	0.00
0.00	0.00	−2.00	4.00	−4.00	4.00	−2.00	0.00
0.00	0.00	0.00	−2.00	4.00	−4.00	4.00	−2.00
0.00	0.00	0.00	0.00	−2.00	4.00	−4.00	6.00
0.00	0.00	0.00	0.00	0.00	−2.00	6.00	−8.00

1D Parallelization(1): Matrix Construction

- Observe: each interval/node contributes to a local matrix

$$a(u,v) \equiv \sum_{j=0}^{j=N} \int_{x_j}^{x_{j+1}} u'v' + \sum_{j=0}^{j=N+1} \Big(\{u'\}_j [v]_j + \underbrace{\{v'\}_j [u]_j}_{\text{symmetric term}} \Big) + \underbrace{\gamma \sum_{j=0}^{N+1} \frac{1}{|I_j|} [u]_j [v]_j}_{\text{penalty term}}$$

Step 3.3. $\frac{\gamma}{|I_j|}[u][v]$

Local matrices

$$\begin{pmatrix} 20.00 & 0.00 \\ 0.00 & 0.00 \end{pmatrix}$$

$$\begin{bmatrix} 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 20.00 & -20.00 & 0.00 \\ 0.00 & -20.00 & 20.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 \end{bmatrix}$$

$$\begin{pmatrix} 0.00 & 0.00 \\ 0.00 & 20.00 \end{pmatrix}$$

	$\phi_0^{(I_0)}$	$\phi_1^{(I_0)}$	$\phi_0^{(I_1)}$	$\phi_1^{(I_1)}$	$\phi_0^{(I_2)}$	$\phi_1^{(I_2)}$	$\phi_0^{(I_2)}$	$\phi_1^{(I_2)}$
	20.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	20.00	−20.00	0.00	0.00	0.00	0.00	0.00
	0.00	−20.00	20.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	20.00	−20.00	0.00	0.00	0.00
	0.00	0.00	0.00	−20.00	20.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	20.00	−20.00	0.00
	0.00	0.00	0.00	0.00	0.00	−20.00	20.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	20.00

1D Parallelization(1): Matrix Construction

- Algorithm
- 1. Initialization: for each processor, receive information of *P* cells

2. Local Matrix Construction

for each processor

for each of *P* cells, calculate the cell matrix

calculate the processor matrix

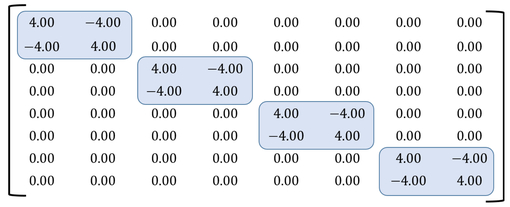
3. Global Matrix Construction

for root processor

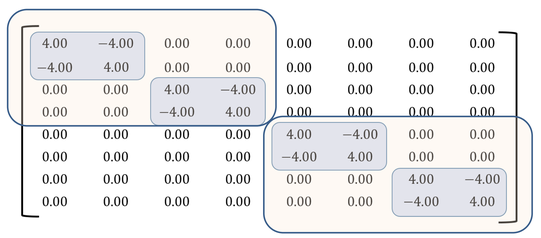
receive information from other processors

calculate the global matrix (*assembly*)

Cell matrices



Processor matrices



Assembly done by *MPI*

1D Parallelization(1): Matrix Construction

- *MPI* Commands: Assembly

- Processors:

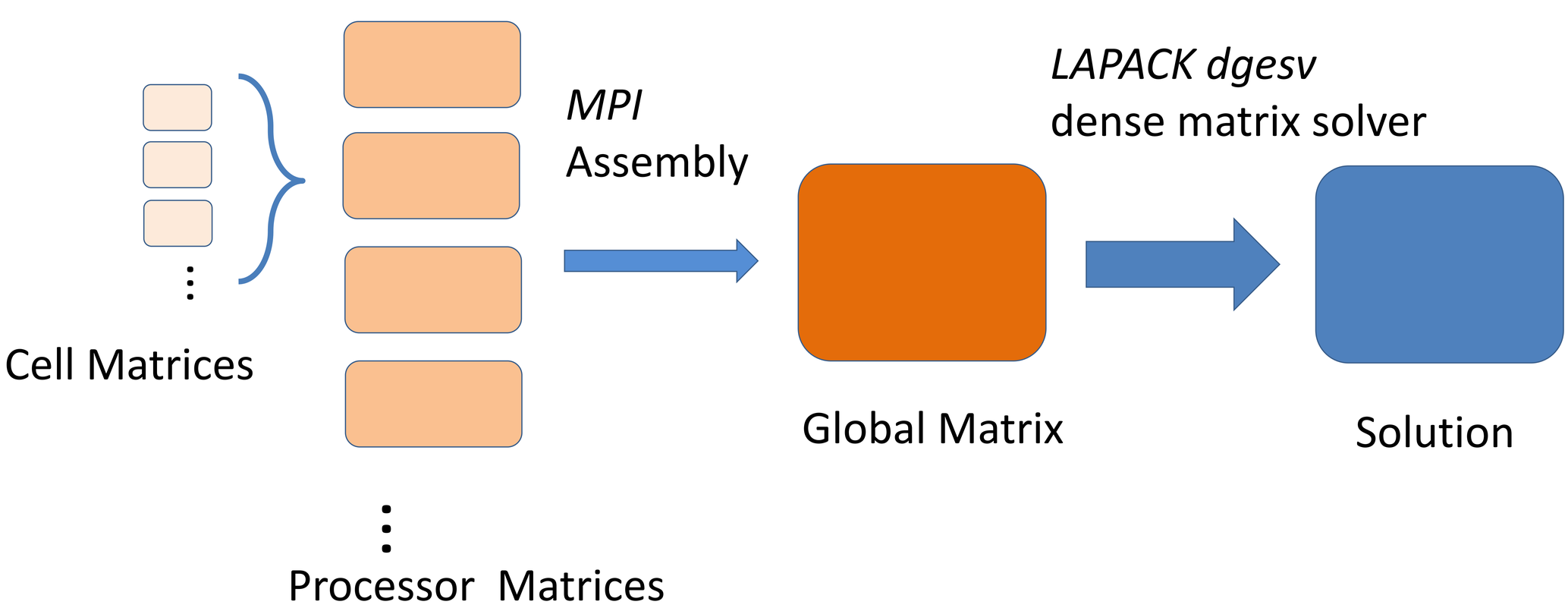
MPI_Send(*pointerToProcessorMatrix, size, MPI_DOUBLE,*
root, senderID, MPI_COMM_WORLD);

Root Processors:

MPI_Recv(*pointerToProcessorMatrix, size, MPI_DOUBLE,*
senderID, tag, MPI_COMM_WORLD, &status);

1D Parallelization(1): Matrix Construction

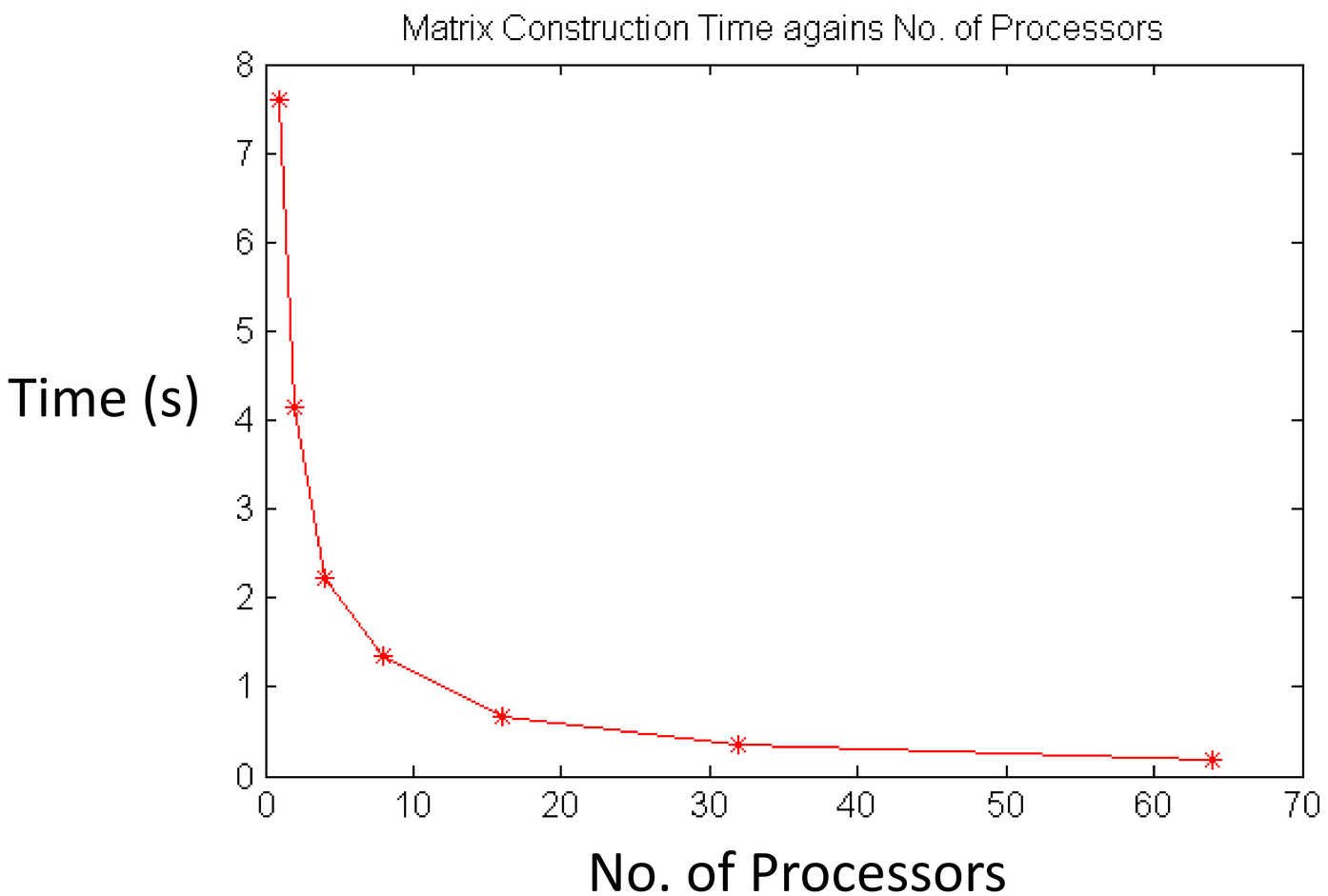
- After Global Matrix Construction
  - Solve linear system
  - C programming: *dgesv* on *LAPACK*
  - LU factorization: dense matrix solver



1D Parallelization(1): Matrix Construction

- Overall Matrix Construction Time decreases with number of processors

No. of Cells: 5,000,000
Degree of freedom: 2



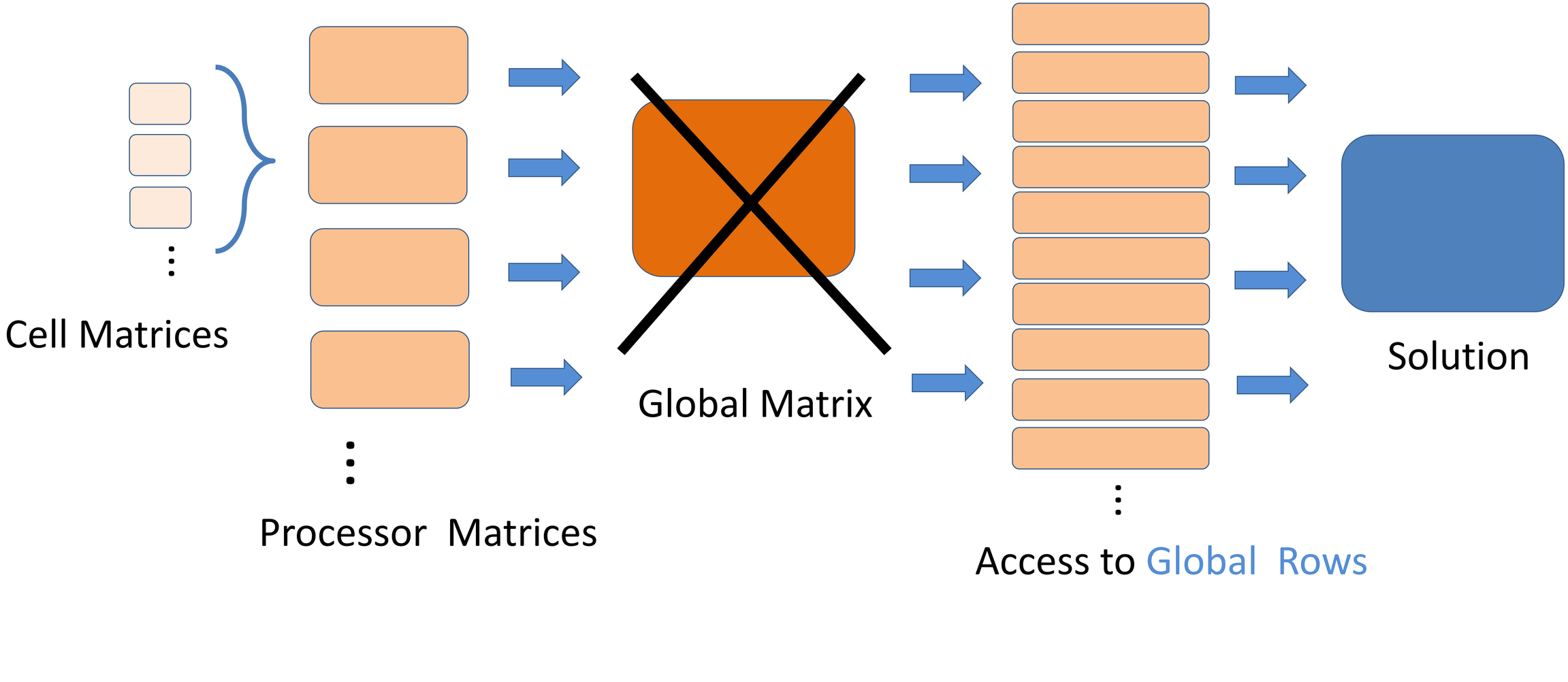
No. of Procs	Time (s)
1	7.60062
2	4.13656s
4	2.22328s
8	1.34502s
16	0.67569s
32	0.36376s
64	0.17482s

"Can we do better?"

1D Parallelization(2): Linear System Solver

- Parallelize the linear system solving process
  - *AztecOO* on *Trilinos*
  - Each processor has access to global rows

AztecOO on Trilinos



1D Parallelization(2): Linear System Solver

- *Trilinos* Commands

1. Define rows owned by each processor

Epetra_Map map(GlobalNumberOfRows, LocalNumberOfRows, GlobalIndicesOfRows, index, comm);

2. Send rows to global matrix position

InsertGlobalValues(GlobalIndexOfRows, NumOfEntries, ValueOfEntries, GlobalColumnIndices);

3. Create solver

```
x = new Epetra_Vector(map);
problem = new Epetra_LinearProblem(GlobalMatrix, x, RHSvector);
AztecOO solver(*problem);
solver.Iterate(MaxNoOfIterations, Tolerance);
```

1D Parallelization(2): Linear System Solver

- 4. Set solver parameters
  - *Solvers*
    - Direct (for dense matrix) or Iterative (for sparse matrix)
    - *LU, GMRES, CG…*
  - *Pre-conditioners*
    - Jacobi, sparse LU factorization, …
    - *ML Library* (Algebraic Multigrid Preconditioning)
      - Large sparse linear system!

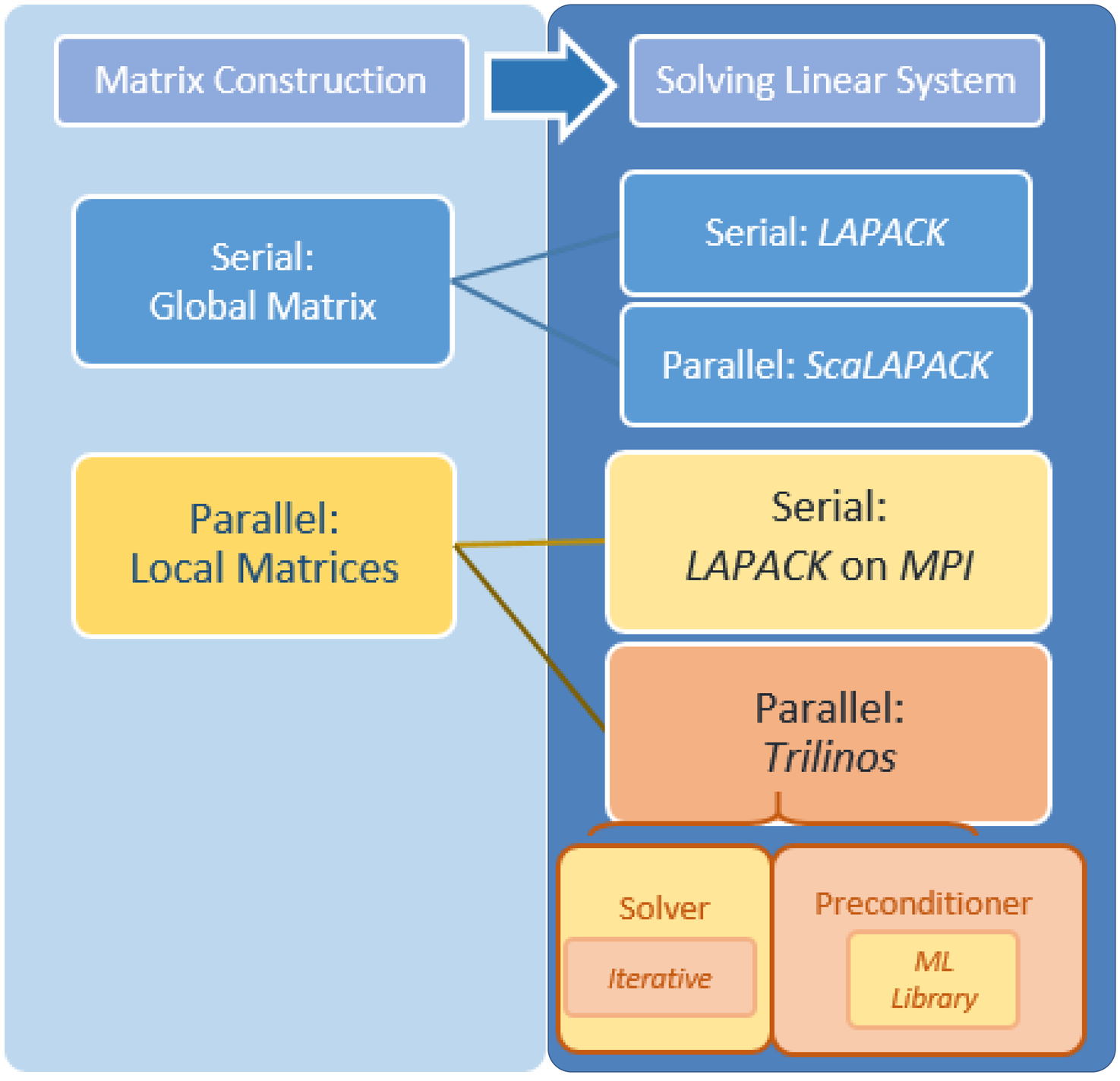
More *Linear Algebra*

1D Parallelization(2): Linear System Solver

- *AztecOO* Commands
  - *solver.SetAztecOption(AZ_solver, SolverType);*
  - *solver.SetAztecOption(AZ_precond, PreCondiType);*

- *ML preconditioner* Commands
  - *ML_Epetra::MultiLevelPreconditioner* MLPrec =*
    new ML_Epetra:: MultiLevelPreconditioner
    (GlobalMatrixPointer, Teuchos::ParameterList MLList, true);
  - *solver.SetPrecOperator(MLPrec)*

1D Parallelization: Overview



1D Parallelization(2): More

- More *Linear Algebra*:
  Check performance of different parameters

7																
8	CG solver time															
9																
10	ilu	7.94739s	2	5.22239s	9	4.88893s	24	5.99693s	40	4.66193s	68	4.130290s	128	4.908537s	249	
11																
12	ML															
13	DD (domain-decomposition method)				Aggregation: Uncoupled			Smoother: Aztec		Coarse type: Amesos-KLU			Increasing		Max Level: 6	
14		11.78661s	2	8.89994s	6	6.21667s	9	5.495537s	10	2.90557s	11	1.33267s	10	1.12753s	21	0.18830s/21
15																
16	DD (domain-decomposition method)				Aggregation: MIS			Smoother: Aztec		Coarse type: Amesos-KLU			Increasing		Max Level: 6	
17		11.77905s	2	8.81326s	6	7.02514s	11	7.13098s	14	4.07430s	17	18.15375s	187	1.12629s	21	
18																
19	DD (domain-decomposition method)				Aggregation: METIS			Smoother: Aztec		Coarse type: Amesos-KLU			Increasing		Max Level: 6	
20		9.43952s	2	7.99520s	8	11.42985s	29	14.72198s	43	13.26475s	81	12.55926s	157	11.19096s	296	
21																
22	DD (domain-decomposition method)				Aggregation: ParMETIS			Smoother: Aztec		Coarse type: Amesos-KLU			Increasing		Max Level: 6	
23		9.03556s	2	8.53104s	10	9.50912s	23	13.79076s	40	12.16044s	74	11.70670s	146	11.971s1s	318	
24																
25	DD (domain-decomposition method)				Aggregation: Uncoupled			Smoother: GaussSeidel		Coarse type: Amesos-KLU			Increasing		Max Level: 6	
26		KILLED	>531	KILLED	>935	KILLED	>1574	--		--			--		--	
27																
28	DD (domain-decomposition method)				Aggregation: Uncoupled			Smoother: Jacobi		Coarse type: Amesos-KLU			Increasing		Max Level: 6	
29		148.23950s	149	82.25089s	150	48.72048s	150	44.53721s	150	22.08759s	150	10.516115s	145	4.984688s	146	0.55250s/62
30																
31	DD (domain-decomposition method)				Aggregation: MIS			Smoother: Jacobi		Coarse type: Amesos-KLU			Increasing		Max Level: 6	
32		142.65031s	149	78.752273s	144	45.62901s	146	43.182687s	145	21.41529s	145	10.97410s	150	4.989396s	145	
33																
34	DD (domain-decomposition method)				Aggregation: METIS			Smoother: Jacobi		Coarse type: Amesos-KLU			Increasing		Max Level: 6	
35		KILLED	>801	KILLED	>1545	KILLED	>2618	KILLED	>2700	KILLED	>5191	KILLED	>10566	KILLED	>22088	
36																
37	DD (domain-decomposition method)				Aggregation: ParMETIS			Smoother: Jacobi		Coarse type: Amesos-KLU			Increasing		Max Level: 6	
38		KILLED	>932	KILLED	>1615	KILLED	>2598	KILLED	>2736	KILLED	>5477	KILLED	>10351	KILLED	>22315	
39																

Uncoupled + Aztec
Good performance

Smoother: GS
not recommeded

Smoother: Jacobi
Increasing perfromance with Procs

Aggregation: METIS
not recommended

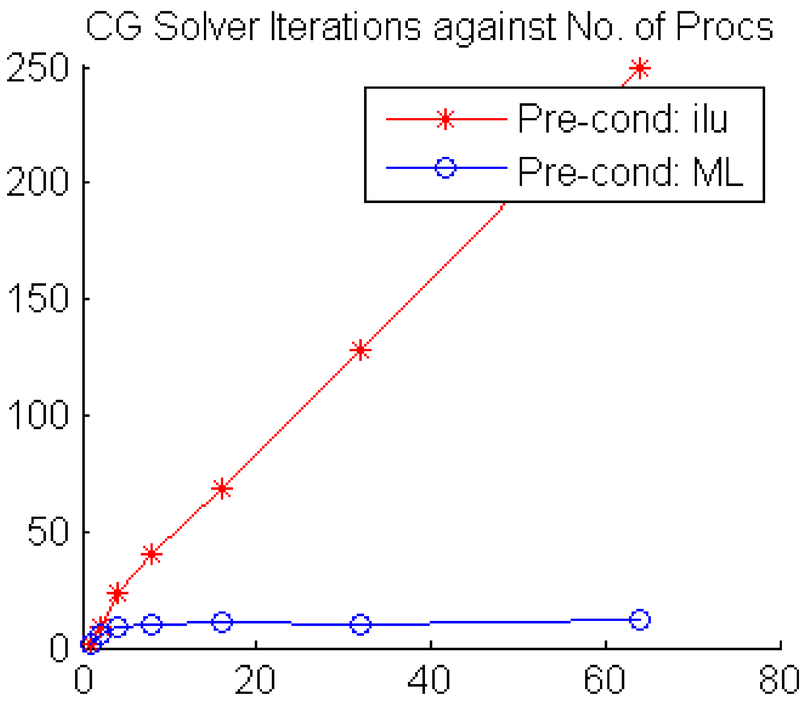
Aggregation: ParMETIS
not recommended

1D Parallelization(2): More

- Best performance found in ML
  - Domain-decomposition method
  - Aggregation: Uncoupled
  - Smoother: Aztec
  - Coarse type: UMFPACK

No. of Cells: 5,000,000
Degree of freedom: 2

	Solver Time/ No. of Iterations	
No. of Procs	CG ML	CG ilu
1	11.81769s/2	7.94730s/2
2	8.87170s/6	5.22223s/9
4	6.22301s/9	4.88893s/24
8	5.53356s/10	4.79693s/40
16	2.91182s/11	4.66193s/68
32	1.33327s/10	4.130290s/128
64	0.72338s/12	4.908537s/249

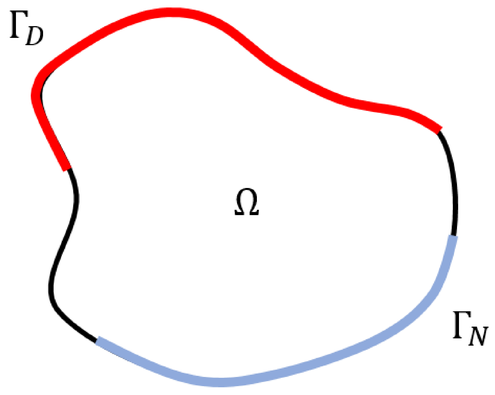


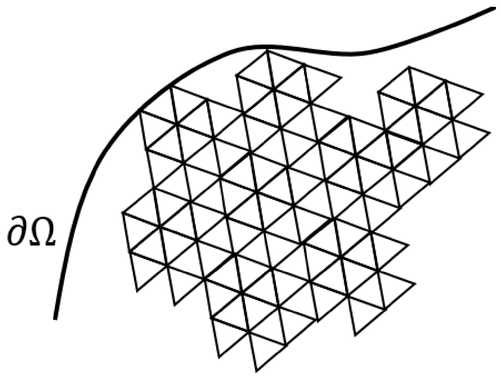
"Can we do better?"

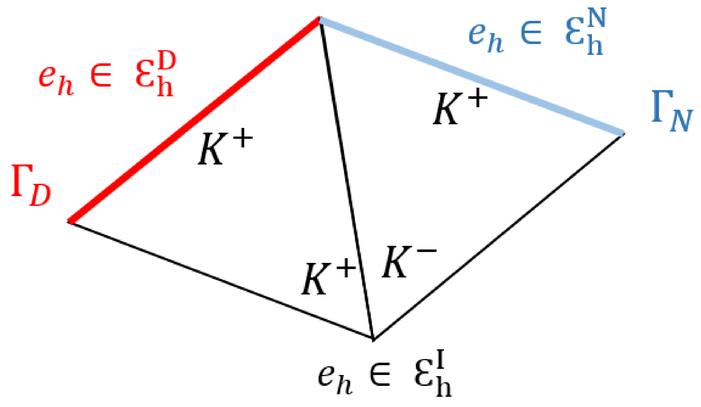
- Overview
- Mathematics behind
- 1D Parallelization
- Future Work
- Acknowledgement

Future Work: 2D DG-FEM

- Implement parallelization on 2D DG-FEM
  - Partition of Domain: Triangles
  - Jump condition over Edges



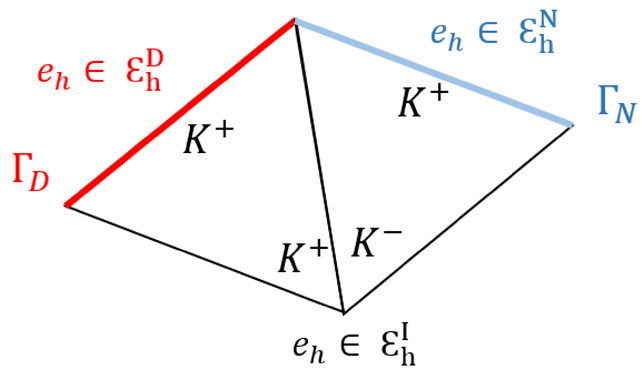




Future Work: 2D DG-FEM

- Implement parallelization on 2D DG-FEM
- Weak form of FEM:

$$-\int_\Omega \triangle u\, v\, dx = \int_\Omega \nabla u \cdot \nabla v\, dx - \int_{\partial\Omega} (\nabla u \cdot)\, v\, ds$$
$$= \int_\Omega \nabla u \cdot \nabla v\, dx - \int_{\partial\Omega} \frac{\partial u}{\partial \mathbf{n}} v\, ds$$
$$= \int_\Omega f v\, dx$$



- Bilinear Function of DG-FEM:

$$a_h(u,v) \equiv \sum_{K\in\mathcal{T}_h} (\nabla u, \nabla v)_K - \sum_{e_h\in\mathcal{E}_h} \left(< \{\partial_n u\}, [v] >_{e_h} + < \{\partial_n v\}, [u] >_{e_h} - \frac{\gamma}{|e_h|} < [u], [v] >_{e_h} \right)$$

Future Work: 2D DG-FEM

$$a_h(u,v) \equiv \sum_{K\in\mathcal{T}_h} (\nabla u, \nabla v)_K - \sum_{e_h\in\mathcal{E}_h} \Big(< \{\partial_n u\}, [v] >_{e_h} + < \{\partial_n v\}, [u] >_{e_h} - \frac{\gamma}{|e_h|} < [u], [v] >_{e_h} \Big)$$

- Correctness given by showing

$$a_h^{\gamma_h}(v,v) \geq c_a \|v\|_{1,h}^2, \quad \forall v \in V^h,$$

where norm defined as

$$\|v\|_{1,h} = \left\{ \sum_{K\in\mathcal{T}_h} \|\nabla v\|_K^2 + \sum_{e_h\in\mathcal{E}_h} |e_h| \, |\{\partial_n v\}|_{e_h}^2 + \sum_{e_h\in\mathcal{E}_h} \frac{\gamma_h}{|e_h|} |[v]|_{e_h}^2 \right\}^{1/2}.$$

Future Work

- … and on 3D DG-FEM

- Adaptive Meshing
  - local refinement

- Application e.g. Chemical Transport Equation

- Overview
- Mathematics behind
- 1D Parallelization
- Future Work
- Acknowledgement

Acknowledgement

This project was sponsored by

Oak Ridge National Laboratory,
Joint Institute for Computational Sciences,
University of Tennessee, Knoxville
and *the Chinese University of Hong Kong*.

Also sincere gratitude to my mentors
Dr. Kwai Wong
and *Dr. Ohannes Karakashian*

Questions