Multi-dimensional Parallel Discontinuous Galerkin Method

Mentors: Dr. Ohannes Karakashian, Dr. Kwai Wong, Michael Wise
Zhe Zhu
The Chinese University of Hong Kong
7/29/2016

Agenda

1. Background
2. How DG works
3. Solve 2D equation
4. Solve 3D Equation
5. Parallelization
6. Future work

Background

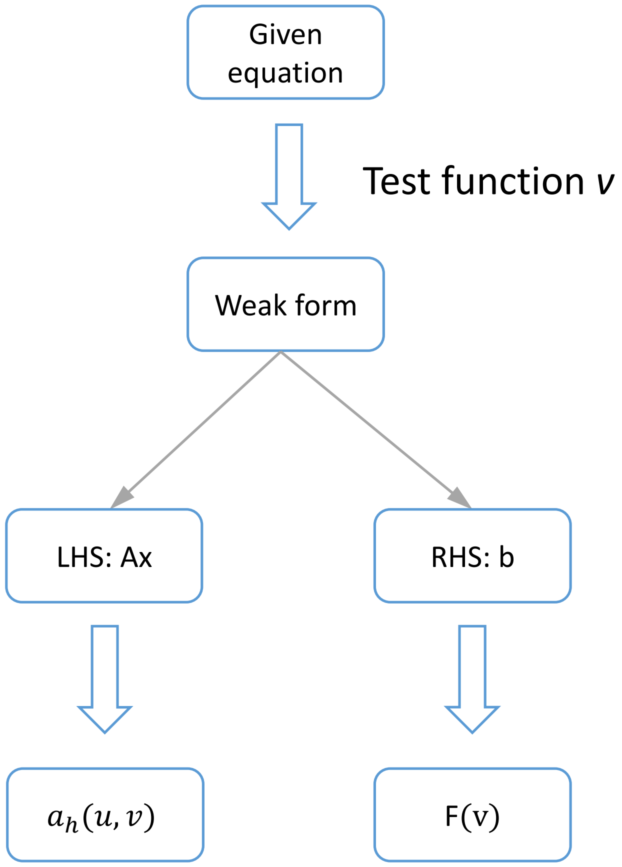
Goal:

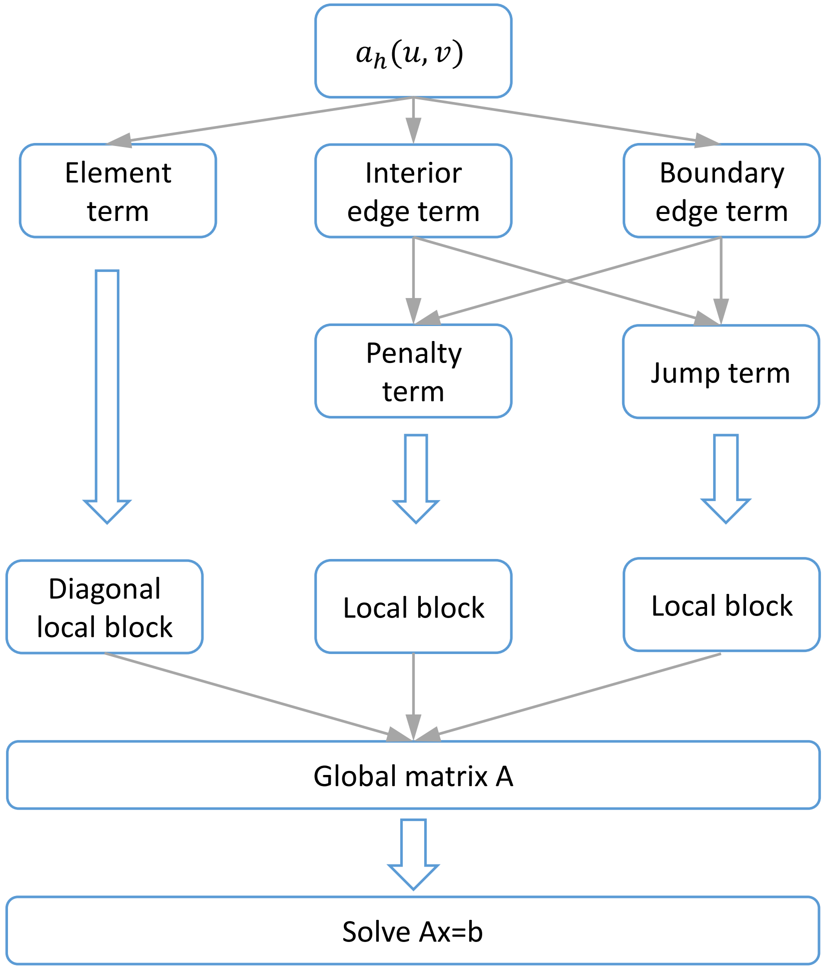
Implement DG-FEM to solve Poisson's equation in parallel on HPC platform

Problem Setting:

$$\begin{cases} -\Delta u = f & in \quad \Omega \\ u = g_d & on \quad \Gamma_D \\ \dfrac{\partial u}{\partial \vec{n}} = g_n & on \quad \Gamma_N \end{cases} \Longrightarrow \quad Ax=b \quad \Longrightarrow \quad \text{Approximate solution of } u$$

How DG works





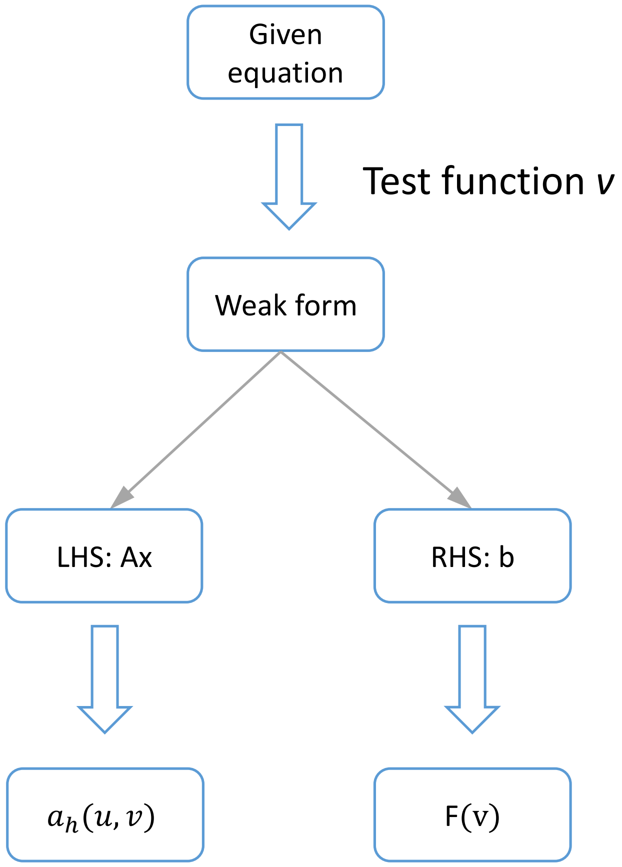
Divided into three parts

Computed in parallel

Combine all local blocks

Use *Trilinos* to finish parallel solving

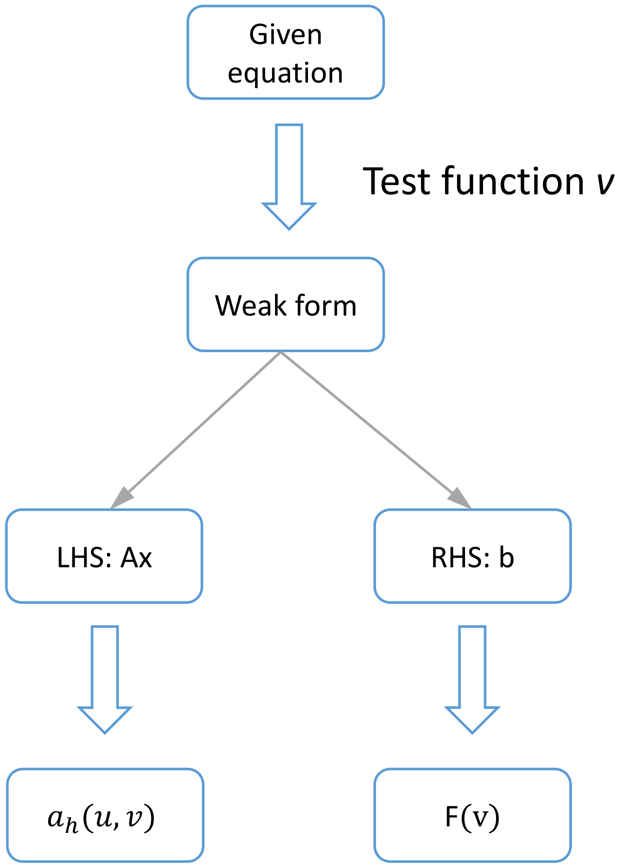
How DG works



$$\begin{cases} -\Delta u = f & in \;\; \Omega \\ u = g_d & on \;\; \Gamma_D \\ \dfrac{\partial u}{\partial \vec{n}} = g_n & on \;\; \Gamma_N \end{cases}$$

$$\begin{aligned} -\int_\Omega \Delta u\, v\, dx &= -\sum_{K\in\mathcal{T}_h}\int_K \Delta u\, v\, dx \\ &= \sum_{K\in\mathcal{T}_h}\int_K \nabla u\cdot\nabla v\, dx - \sum_{K\in\mathcal{T}_h}\int_{\partial K}\frac{\partial u}{\partial \mathbf{n}} v\, ds \\ &= \sum_{K\in\mathcal{T}_h}\int_K \nabla u\cdot\nabla v\, dx - \sum_{e_h\in\mathcal{E}_h^D}\int_{e_h}\frac{\partial u}{\partial \mathbf{n}} v\, ds - \sum_{e_h\in\mathcal{E}_h^N}\int_{e_h}\frac{\partial u}{\partial \mathbf{n}} v\, ds \\ &\quad - \sum_{e_h\in\mathcal{E}_h^I}\int_{e_h}\left(\frac{\partial u^+}{\partial \mathbf{n}n^+} v^+ + \frac{\partial u^-}{\partial \mathbf{n}n^-} v^-\right) ds \\ &= \int_\Omega f\, v\, dx \end{aligned}$$

How DG works



$$\begin{cases} -\Delta u = f & in \ \ \Omega \\ u = g_d & on \ \ \Gamma_D \\ \dfrac{\partial u}{\partial \vec{n}} = g_n & on \ \ \Gamma_N \end{cases}$$

Bilinear Function for Stiffness Matrix:

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

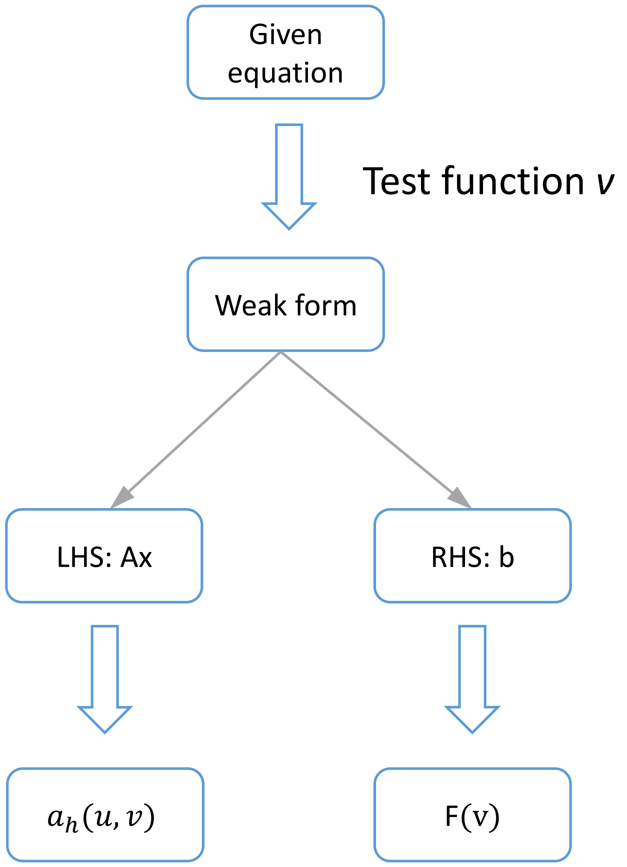
$$- \sum_{e_h \in \mathcal{E}_h^D} \left(< \partial_n u, v >_{e_h} + < \partial_n v, u >_{e_h} - \frac{\gamma}{|e_h|} < u, v >_{e_h} \right)$$

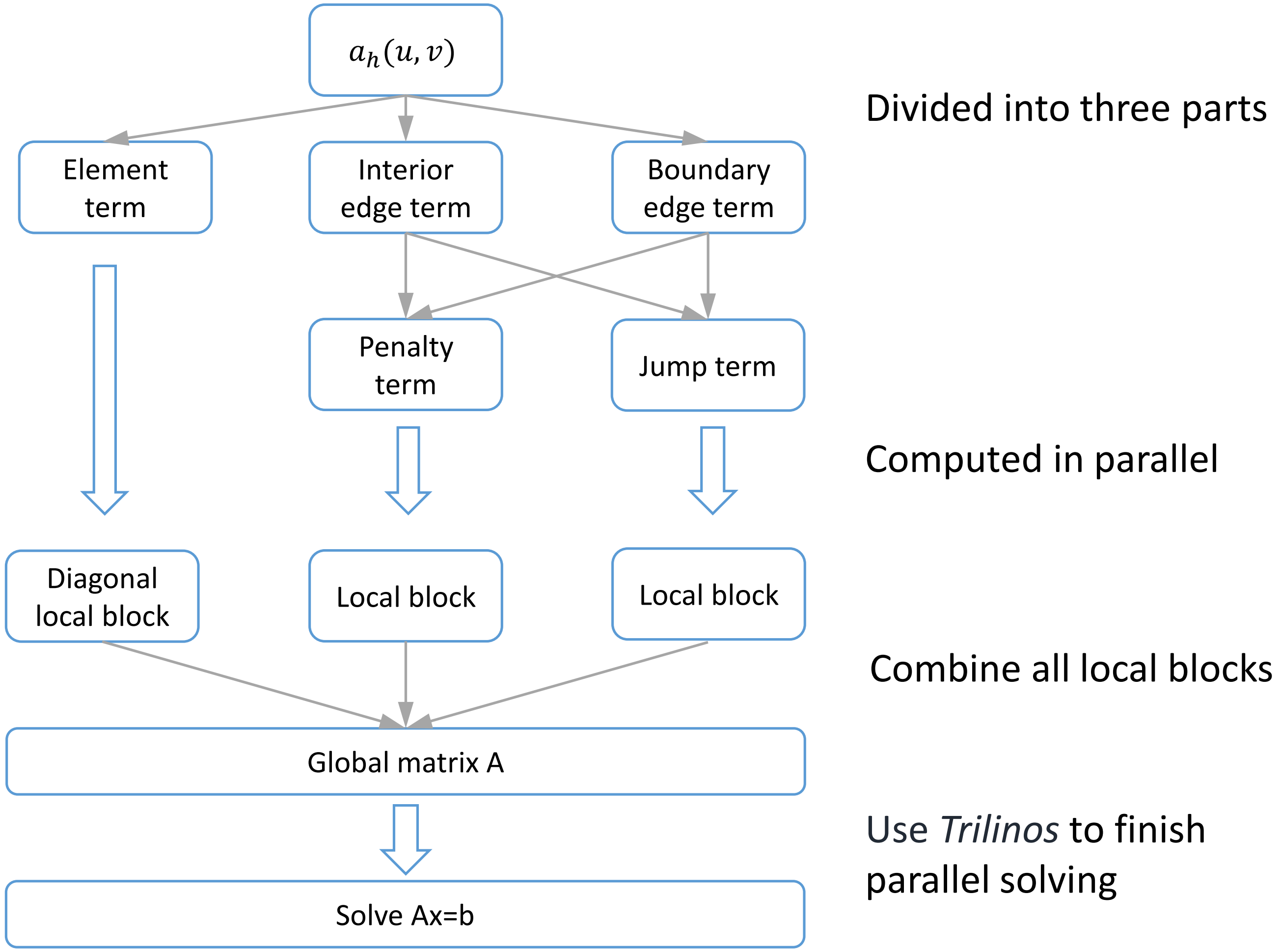
:element term : jump term :penalty term

Solving Linear System:

$$\sum_{j=1}^{j=M} \underbrace{a(\phi_j, \phi_i)}_{S_{ij}} \alpha_j = \underbrace{\int f \, \phi_i}_{r_i} + \text{symmetric term} + \text{penalty term}$$

How DG works


Given equation
Test function *v*
Weak form
LHS: Ax
RHS: b
$a_h(u, v)$
F(v)


$a_h(u, v)$
Divided into three parts
Element term
Interior edge term
Boundary edge term
Penalty term
Jump term
Computed in parallel
Diagonal local block
Local block
Local block
Combine all local blocks
Global matrix A
Use *Trilinos* to finish parallel solving
Solve Ax=b

How DG works

Bilinear Function for Stiffness Matrix:

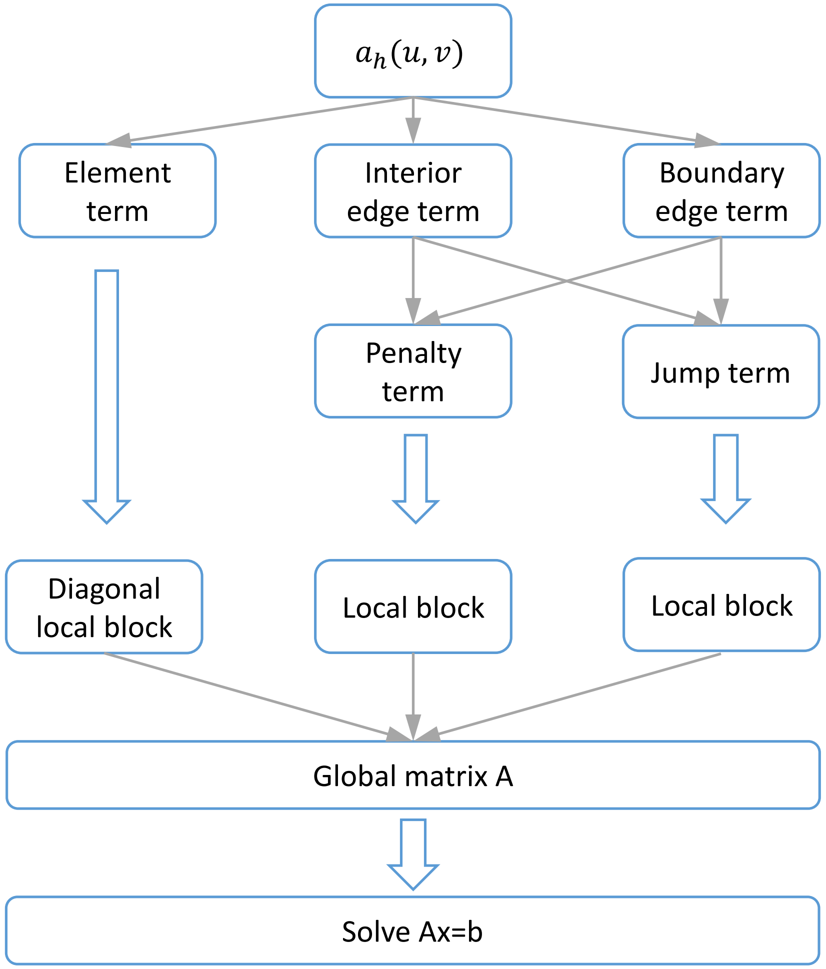
$$a_h(u,v) \equiv \underbrace{\sum_{K\in\mathcal{T}_h} (\nabla u, \nabla v)_K}_{} - \sum_{e_h \in \mathcal{E}_h^I} \left(< \{\partial_n u\}, [v] >_{e_h} + < \{\partial_n v\}, [u] >_{e_h} - \frac{\gamma}{|e_h|} < [u], [v] >_{e_h} \right)$$

$$- \sum_{e_h \in \mathcal{E}_h^D} \left(< \partial_n u, v >_{e_h} + < \partial_n v, u >_{e_h} - \frac{\gamma}{|e_h|} < u, v >_{e_h} \right)$$

:element term : jump term :penalty term

Solving Linear System:

$$\sum_{j=1}^{j=M} \underbrace{a(\phi_j, \phi_i)}_{S_{ij}} \alpha_j = \underbrace{\int f \, \phi_i}_{r_i} + \text{symmetric term} + \text{penalty term}$$



Divided into three parts

Computed in parallel

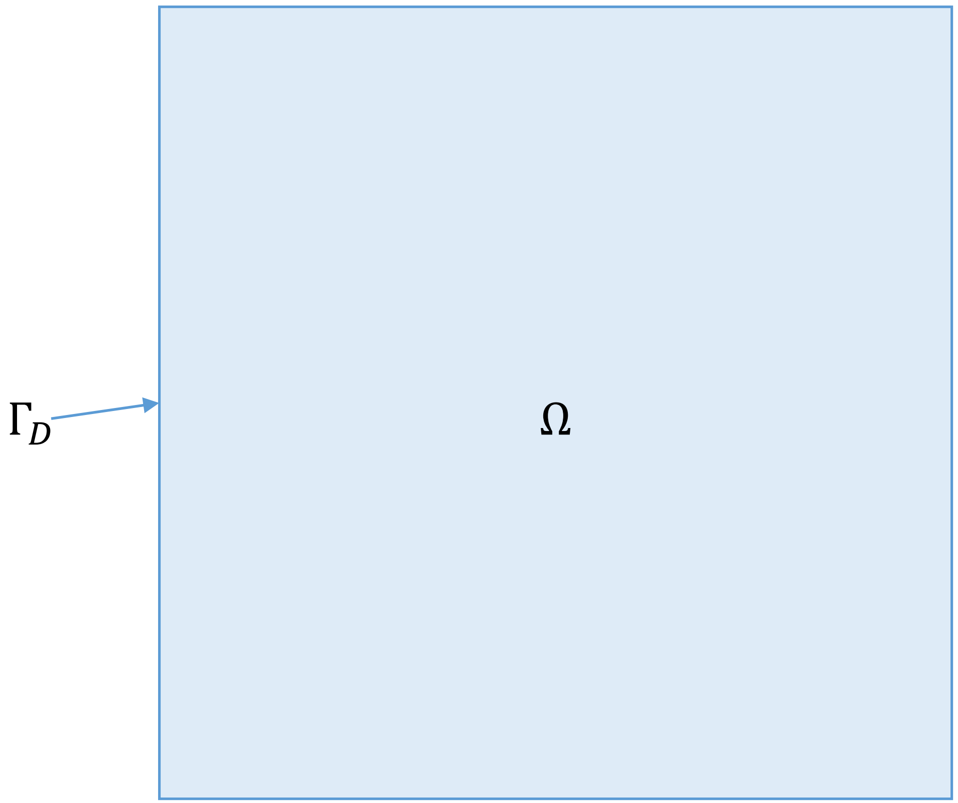
Combine all local blocks

Use *Trilinos* to finish parallel solving

Solve 2D Equation

$$\begin{cases} -\Delta u = f & in \quad \Omega \\ u = g_d & on \quad \Gamma_D \\ \dfrac{\partial u}{\partial \vec{n}} = g_n & on \quad \Gamma_N \end{cases}$$

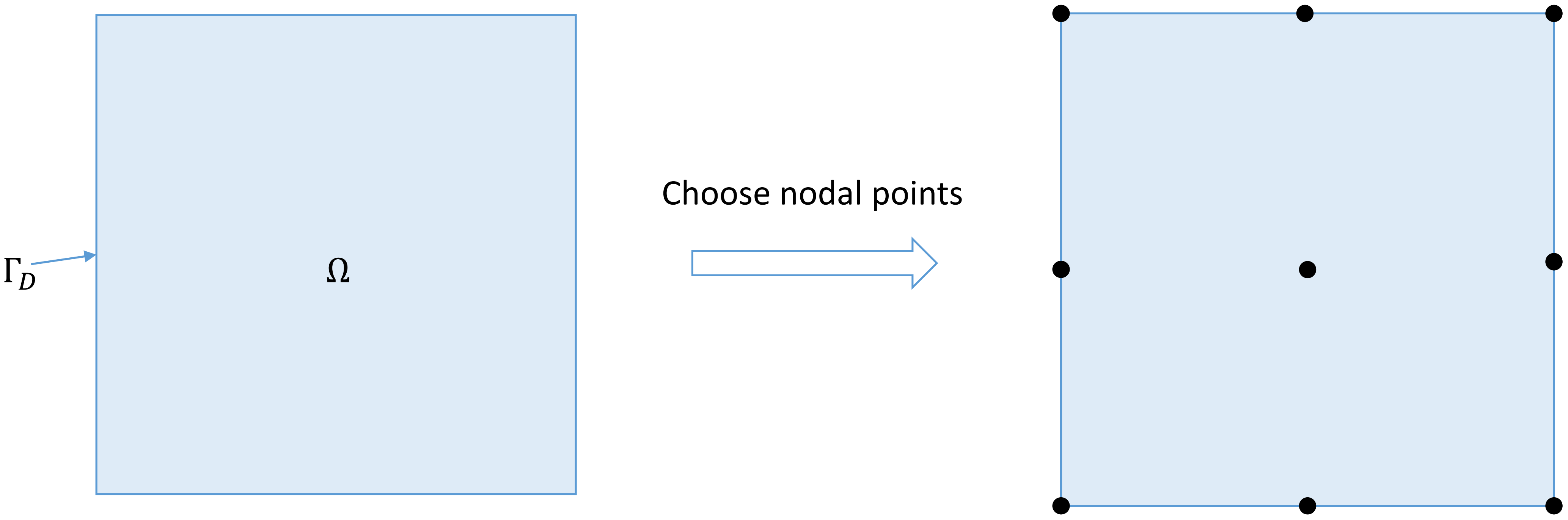
Where f, g_d, g_n is known



Sample 2D Mesh

- We want to solve u through numerical method
- We want to get an approximation of u on the domain.

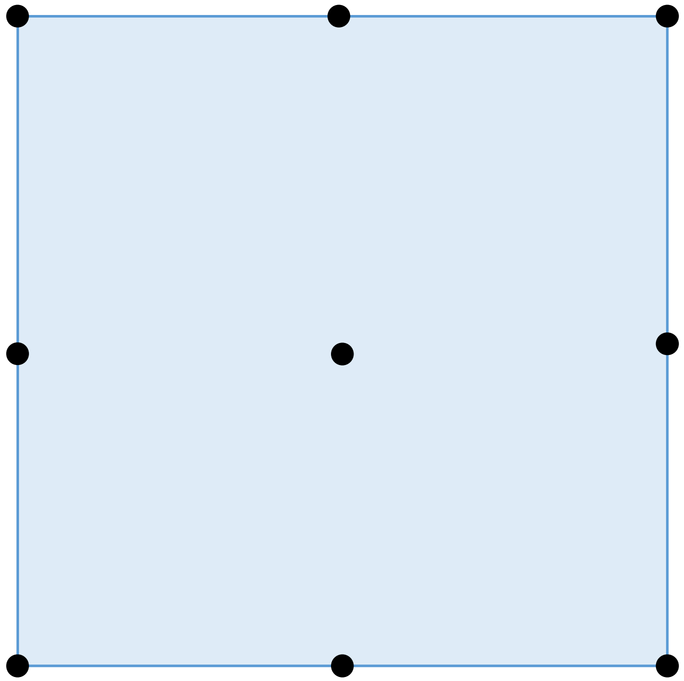
Solve 2D Equation



- By calculating the value of u on these nodal points, we can get the approximation of u on Ω
- In general, the more nodal points you choose, the higher accuracy of the approximation you get

Solve 2D Equation

How to get the approximation of *u* on these nodal points ?



1. Find a set of independent basic functions $\{ v_j\}$
2. Try to use the linear combination to get the approximate value on nodal points which can fit into the exact value.
3. Represent *u* as a linear combination of those basic functions

$$u \quad = \sum_j \hat{u}_j \, v_j$$

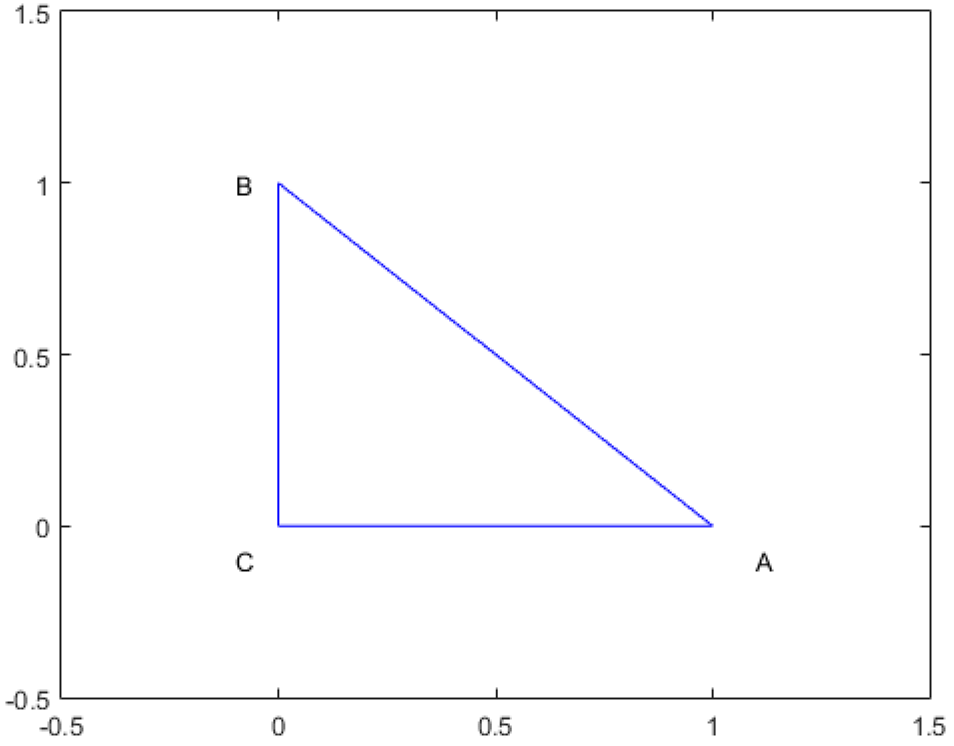
Solve 2D Equation

The two problem we need to solve:

1. How to choose the basic functions?

2. How to represent *u* as a linear combination of basic functions?

Solve 2D Equation



Master cell

lagrange interpolation

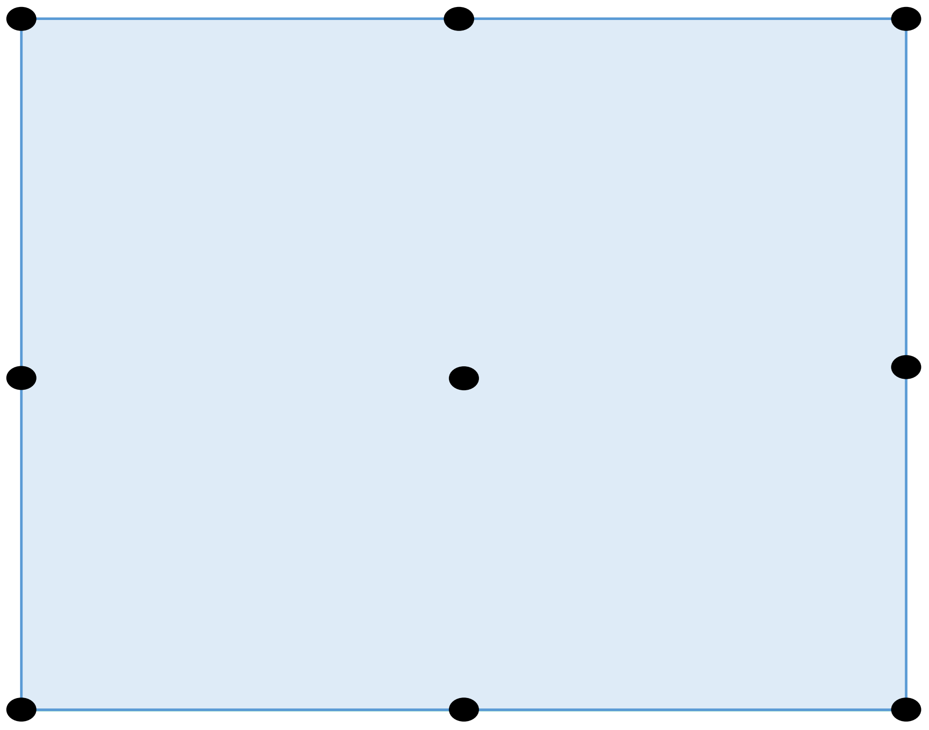
$\varphi_A =$ x

$\varphi_B =$ y

$\varphi_C =$ 1-x-y

Solve 2D Equation

Divide the mesh into several elements



$\varphi_A = x$

$\varphi_B = y$

$\varphi_C = 1-x-y$

φ_A^0

φ_B^0

φ_C^0

φ_A^1

φ_B^1

φ_C^1

$\vdots$

φ_A^7

φ_B^7

φ_C^7

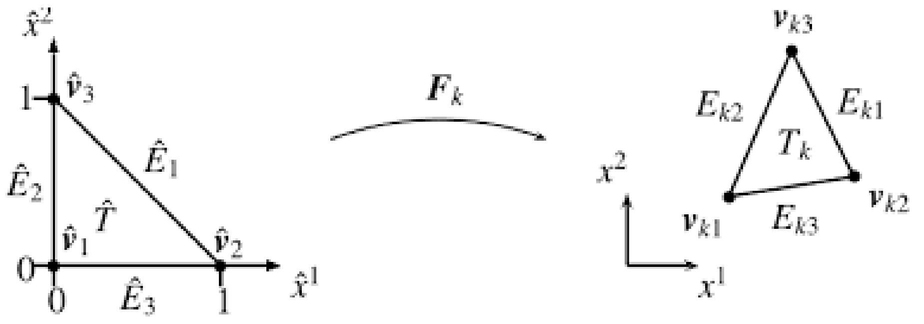
Solve 2D Equation

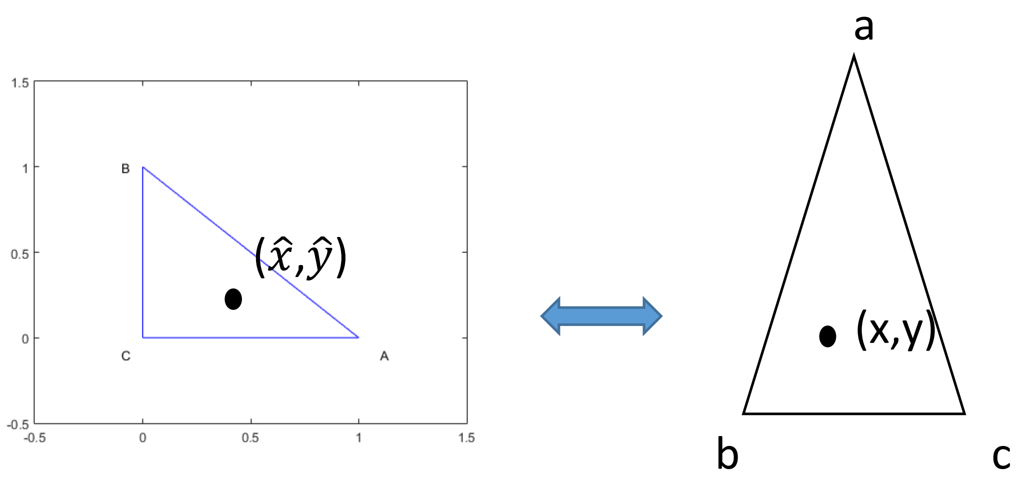
Use map to do integration in the master cell

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_a - x_c & x_b - x_c \\ y_a - y_c & y_b - y_c \end{pmatrix} * \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} + \begin{pmatrix} x_c \\ y_c \end{pmatrix}$$

$$J_{(\hat{x},\hat{y})} = det\begin{pmatrix} x_a - x_c & x_b - x_c \\ y_a - y_c & y_b - y_c \end{pmatrix} = \begin{pmatrix} \frac{\partial x}{\partial \hat{x}} & \frac{\partial x}{\partial \hat{y}} \\ \frac{\partial y}{\partial \hat{x}} & \frac{\partial y}{\partial \hat{y}} \end{pmatrix}$$

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \frac{1}{|J_{(\hat{x},\hat{y})}|} \begin{pmatrix} y_a - y_c & x_c - x_b \\ y_c - y_a & x_b - x_c \end{pmatrix} * \begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix}$$





Solve 2D Equation

Bilinear Function for Stiffness Matrix:

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

$$- \sum_{e_h\in\mathcal{E}_h^D} \left(< \partial_n u, v >_{e_h} + < \partial_n v, u >_{e_h} - \frac{\gamma}{|e_h|} < u, v >_{e_h} \right)$$

:element term : jump term :penalty term

Solving Linear System:

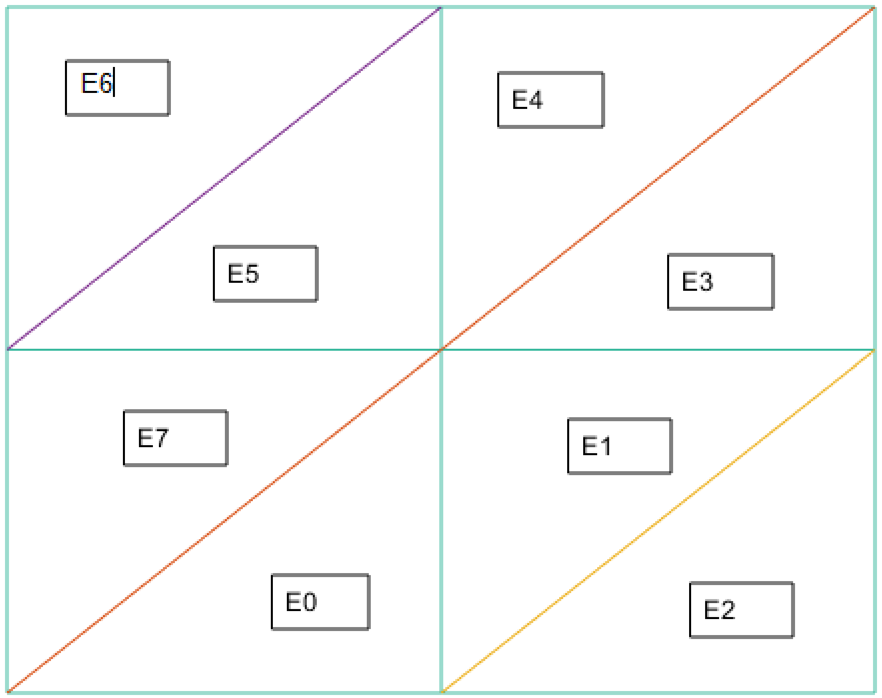
$$\sum_{j=1}^{j=M} \underbrace{a(\phi_j, \phi_i)}_{S_{ij}} \alpha_j = \underbrace{\int f \, \phi_i}_{r_i} + \text{symmetric term} + \text{penalty term}$$

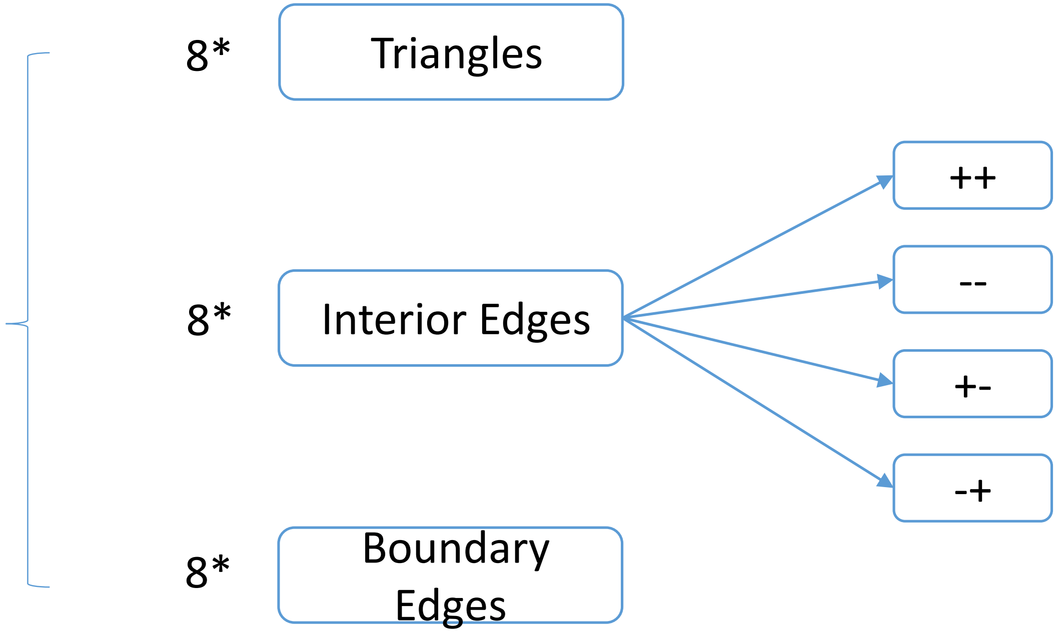
Solve 2D Equation

$$
\begin{array}{c|c}
 & \begin{matrix} \phi_0^{(I_0)} & \phi_1^{(I_0)} & \phi_2^{(I_0)} & \phi_0^{(I_1)} & \phi_1^{(I_1)} & \phi_2^{(I_1)} \end{matrix} \\
\begin{matrix} \phi_0^{(I_0)} \\ \phi_1^{(I_0)} \\ \phi_2^{(I_0)} \\ \phi_0^{(I_1)} \\ \phi_2^{(I_1)} \\ \phi_1^{(I_1)} \end{matrix} &
\left[\begin{array}{ccc|ccc}
a(\phi_0^{(E_0)},\phi_0^{(E_0)}) & a(\phi_1^{(E_0)},\phi_0^{(E_0)}) & a(\phi_2^{(E_0)},\phi_0^{(E_0)}) & a(\phi_0^{(E_1)},\phi_0^{(E_0)}) & a(\phi_1^{(E_1)},\phi_0^{(E_0)}) & a(\phi_2^{(E_1)},\phi_0^{(E_0)}) & \cdots \\
a(\phi_0^{(E_0)},\phi_1^{(E_0)}) & a(\phi_1^{(E_0)},\phi_1^{(E_0)}) & a(\phi_2^{(E_0)},\phi_1^{(E_0)}) & a(\phi_0^{(E_1)},\phi_1^{(E_0)}) & a(\phi_1^{(E_1)},\phi_1^{(E_0)}) & a(\phi_2^{(E_1)},\phi_1^{(E_0)}) & \\
a(\phi_0^{(E_0)},\phi_2^{(E_0)}) & a(\phi_1^{(E_0)},\phi_2^{(E_0)}) & a(\phi_2^{(E_0)},\phi_2^{(E_0)}) & a(\phi_0^{(E_1)},\phi_2^{(E_0)}) & a(\phi_1^{(E_1)},\phi_2^{(E_0)}) & a(\phi_2^{(E_1)},\phi_2^{(E_0)}) & \\
a(\phi_0^{(E_0)},\phi_0^{(E_1)}) & a(\phi_1^{(E_0)},\phi_0^{(E_1)}) & a(\phi_2^{(E_0)},\phi_0^{(E_1)}) & \ddots & & & \\
a(\phi_0^{(E_0)},\phi_1^{(E_1)}) & a(\phi_1^{(E_0)},\phi_1^{(E_1)}) & a(\phi_2^{(E_0)},\phi_1^{(E_1)}) & & & & \\
a(\phi_0^{(E_0)},\phi_2^{(E_1)}) & a(\phi_1^{(E_0)},\phi_2^{(E_1)}) & a(\phi_2^{(E_0)},\phi_2^{(E_1)}) & & & & \\
\vdots & & & & & &
\end{array}\right]
\end{array}
\begin{bmatrix} \hat{u}_0^{(I_0)} \\ \hat{u}_1^{(I_0)} \\ \hat{u}_2^{(I_0)} \\ \hat{u}_0^{(I_1)} \\ \hat{u}_1^{(I_1)} \\ \hat{u}_2^{(I_0)} \\ \vdots \end{bmatrix} = \cdots
$$

Solve 2D Equation

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





Solve 2D Equation

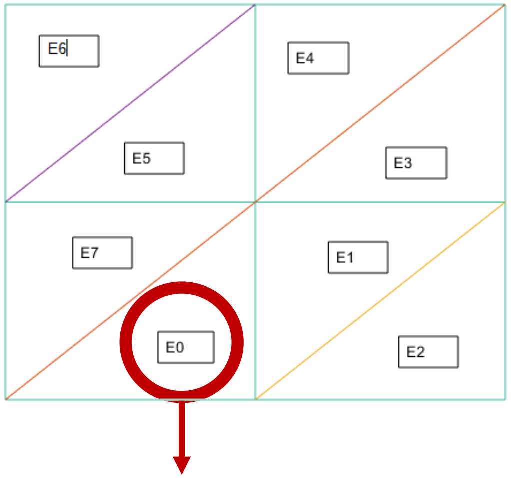
$$a_h(u,v) \equiv \sum_{K\in\mathcal{T}_h} (\nabla u, \nabla v)_K - \sum_{e_h\in\mathcal{E}_h^I} \Big(<\{\partial_n u\},[v]>_{e_h} + <\{\partial_n v\},[u]>_{e_h} - \frac{\gamma}{|e_h|} <[u],[v]>_{e_h} \Big) - \sum_{e_h\in\mathcal{E}_h^D} \Big(<\partial_n u, v>_{e_h} + <\partial_n v, u>_{e_h} - \frac{\gamma}{|e_h|}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

$$\begin{aligned}
\int_k \nabla u \nabla v ds &= \int\int \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\right)\left(\frac{\partial v}{\partial x}, \frac{\partial v}{\partial y}\right) dxdy \\
&= \int\int \left(\frac{\partial u}{\partial x}\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\frac{\partial v}{\partial y}\right) dxdy \\
&= \int\int \left[\left(\frac{\partial u}{\partial \eta}\frac{\partial \eta}{\partial x} + \frac{\partial u}{\partial \xi}\frac{\partial \xi}{\partial x}\right)\left(\frac{\partial v}{\partial \eta}\frac{\partial \eta}{\partial x} + \frac{\partial v}{\partial \xi}\frac{\partial \xi}{\partial x}\right) + \left(\frac{\partial u}{\partial \eta}\frac{\partial \eta}{\partial y} + \frac{\partial u}{\partial \xi}\frac{\partial \xi}{\partial y}\right)\left(\frac{\partial v}{\partial \eta}\frac{\partial \eta}{\partial y} + \frac{\partial v}{\partial \xi}\frac{\partial \xi}{\partial y}\right)\right] |J_{(\eta,\xi)}| d\eta d\xi \\
&= \int\int F(\eta,\xi) d\eta d\xi \\
\text{(Gaussian quadrature)} &= \sum w_i F(\eta_i, \xi_i)
\end{aligned}$$

Also, since we have the map, the value of $\frac{\partial \eta}{\partial x}, \frac{\partial \eta}{\partial y}, \frac{\partial \xi}{\partial x}$ and $\frac{\partial \xi}{\partial y}$ are known.

Solve 2D Equation

For triangles:



0.5000	-0.5000	0
-0.5000	1.0000	-0.5000
0	-0.5000	0.5000

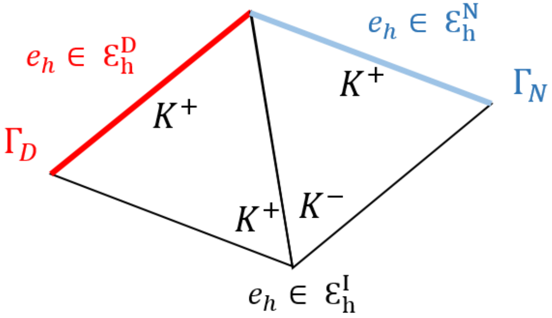
Element-element matrix:

Columns 1 through 12

0.5000	-0.5000	0	0	0	0	0	0	0	0	0	0
-0.5000	1.0000	-0.5000	0	0	0	0	0	0	0	0	0
0	-0.5000	0.5000	0	0	0	0	0	0	0	0	0
0	0	0	0.5000	-0.5000	0	0	0	0	0	0	0
0	0	0	-0.5000	1.0000	-0.5000	0	0	0	0	0	0
0	0	0	0	-0.5000	0.5000	0	0	0	0	0	0
0	0	0	0	0	0	0.5000	-0.5000	0	0	0	0
0	0	0	0	0	0	-0.5000	1.0000	-0.5000	0	0	0
0	0	0	0	0	0	0	-0.5000	0.5000	0	0	0
0	0	0	0	0	0	0	0	0	0.5000	-0.5000	0
0	0	0	0	0	0	0	0	0	-0.5000	1.0000	-0.5000
0	0	0	0	0	0	0	0	0	0	-0.5000	0.5000
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

Solve 2D Equation

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



- $a(\phi_j^{k^+}, \phi_i^{k^+})$: the (i,j) element of (k^+, k^+) block.
- $a(\phi_j^{k^-}, \phi_i^{k^-})$: the (i,j) element of (k^-, k^-) block.
- $a(\phi_j^{k^+}, \phi_i^{k^-})$: the (i,j) element of (k^-, k^+) block
- $a(\phi_j^{k^-}, \phi_i^{k^+})$: the (i,j) element of (k^+, k^-) block

Solve 2D Equation

$$I(\phi_j^{k^+},\phi_i^{k^+}) = -\sum_e \Big(\frac{1}{2} < \partial_n(\phi_j^{k^+})^+ + \partial_n(\phi_j^{k^+})^-), (\phi_i^{k^+})^+ - (\phi_i^{k^+})^- >_e$$

$$+\frac{1}{2} < \partial_n(\phi_i^{k^+})^+ + \partial_n(\phi_i^{k^+})^-), (\phi_j^{k^+})^+ - (\phi_j^{k^+})^- >_e$$

$$-\frac{\gamma}{|e_n|} < (\phi_j^{k^+})^+ - (\phi_j^{k^+})^-, (\phi_i^{k^+})^+ - (\phi_i^{k^+})^- >_e \Big)$$

$$= -\frac{1}{2} < \partial_n(\phi_j^{k^+})^+, (\phi_i^{k^+})^+ >_e - \frac{1}{2} < \partial_n(\phi_i^{k^+})^+, (\phi_j^{k^+})^+ >_e + \frac{\gamma}{|e_n|} < (\phi_j^{k^+})^+, (\phi_i^{k^+})^+ >_e$$

$$I(\phi_j^{k^+},\phi_i^{k^-}) = -\sum_e \Big(\frac{1}{2} < \partial_n(\phi_j^{k^+})^+ + \partial_n(\phi_j^{k^+})^-), (\phi_i^{k^-})^+ - (\phi_i^{k^-})^- >_e$$

$$+\frac{1}{2} < \partial_n(\phi_i^{k^-})^+ + \partial_n(\phi_i^{k^-})^-), (\phi_j^{k^+})^+ - (\phi_j^{k^+})^- >_e$$

$$-\frac{\gamma}{|e_n|} < (\phi_j^{k^+})^+ - (\phi_j^{k^+})^-, (\phi_i^{k^-})^+ - (\phi_i^{k^-})^- >_e \Big)$$

$$= -\frac{1}{2} < \partial_n(\phi_j^{k^+})^+, (\phi_i^{k^-})^- >_e - \frac{1}{2} < \partial_n(\phi_i^{k^-})^-, (\phi_j^{k^+})^+ >_e - \frac{\gamma}{|e_n|} < (\phi_j^{k^+})^+, (\phi_i^{k^-})^- >_e$$

Solve 2D Equation

$$\frac{1}{2}\int_{E_i} \nabla u^+ \mathbf{n} v^+ ds = \frac{1}{2}\frac{\text{area of } E_0}{\text{area of master cell}}\int_{E_i} \nabla \hat{u} \frac{1}{|J_{(\hat{x},\hat{y})}|}\begin{pmatrix} y_a - y_c & x_c - x_b \\ y_c - y_a & x_b - x_c \end{pmatrix} \cdot \mathbf{n} \hat{v} d\hat{s}$$

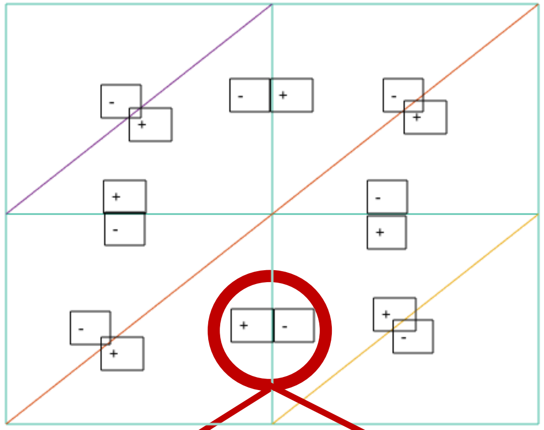
We use Gaussian quadrature to do the integration

- In this example, the function is linear, so we only need one quadrature point to get the answer.
- When calculate penalty term, we will need at least two quadrature point.
- The match of the more than one pair of quadrature points is a problem.

Solve 2D Equation

Get the right number for each blocks and combine those blocks into the global matrix

For edge (++ case)



0	−0.1250	0.1250
−0.1250	−0.2500	0.1250
0.1250	0.1250	0

4	1	1
1	2	0
1	0	2

Columns 1 through 12

3.5000	1.2500	0.7500	0	0	0	0	0	0	0	0	0
1.2500	1.5000	0.7500	0	0	0	0	0	0	0	0	0
0.7500	0.7500	1.5000	0	0	0	0	0	0	0	0	0
0	0	0	1.5000	0.7500	0.7500	0	0	0	0	0	0
0	0	0	0.7500	1.5000	1.2500	0	0	0	0	0	0
0	0	0	0.7500	1.2500	3.5000	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1.5000	0.5000	0.5000
0	0	0	0	0	0	0	0	0	0.5000	0	0.5000
0	0	0	0	0	0	0	0	0	0.5000	0.5000	1.5000
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

Solve 2D Equation

Columns 1 through 12

4.0000	1.2500	1.2500	-2.0000	-0.7500	-0.2500	0	0	0	0	0	0
1.2500	3.5000	0.7500	-0.7500	-1.5000	-0.2500	0	0	0	0	0	0
1.2500	0.7500	4.0000	-0.2500	-0.2500	0	0	0	0	0	0	0
-2.0000	-0.7500	-0.2500	4.0000	1.0000	1.0000	-1.5000	-0.5000	-0.5000	0	-0.2500	-0.2500
-0.7500	-1.5000	-0.2500	1.0000	4.0000	1.0000	-0.5000	0	-0.5000	-0.2500	-1.5000	-0.7500
-0.2500	-0.2500	0	1.0000	1.0000	4.0000	-0.5000	-0.5000	-1.5000	-0.2500	-0.7500	-2.0000
0	0	0	-1.5000	-0.5000	-0.5000	4.0000	1.0000	1.5000	0	0	0
0	0	0	-0.5000	0	-0.5000	1.0000	3.0000	1.0000	0	0	0
0	0	0	-0.5000	-0.5000	-1.5000	1.5000	1.0000	4.0000	0	0	0
0	0	0	0	-0.2500	-0.2500	0	0	0	4.0000	0.7500	1.2500
0	0	0	-0.2500	-1.5000	-0.7500	0	0	0	0.7500	3.5000	1.2500
0	0	0	-0.2500	-0.7500	-2.0000	0	0	0	1.2500	1.2500	4.0000
0	0	0	0	0	0	0	0	0	-1.5000	-0.5000	-0.5000
0	0	0	0	0	0	0	0	0	-0.5000	0	-0.5000
0	0	0	0	0	0	0	0	0	-0.5000	-0.5000	-1.5000
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
-1.5000	-0.5000	-0.5000	0	0	0	0	0	0	0	0	0
-0.5000	0	-0.5000	0	0	0	0	0	0	0	0	0
-0.5000	-0.5000	-1.5000	0	0	0	0	0	0	0	0	0

Columns 13 through 24

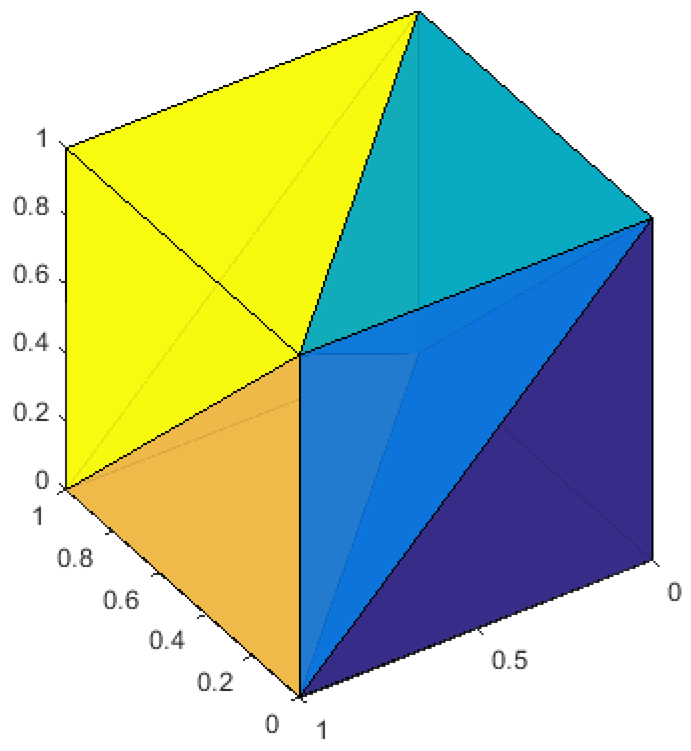
0	0	0	0	0	0	0	0	0	-1.5000	-0.5000	-0.5000
0	0	0	0	0	0	0	0	0	-0.5000	0	-0.5000
0	0	0	0	0	0	0	0	0	-0.5000	-0.5000	-1.5000
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
-1.5000	-0.5000	-0.5000	0	0	0	0	0	0	0	0	0
-0.5000	0	-0.5000	0	0	0	0	0	0	0	0	0
-0.5000	-0.5000	-1.5000	0	0	0	0	0	0	0	0	0
4.0000	1.2500	1.2500	-2.0000	-0.7500	-0.2500	0	0	0	0	0	0
1.2500	3.5000	0.7500	-0.7500	-1.5000	-0.2500	0	0	0	0	0	0
1.2500	0.7500	4.0000	-0.2500	-0.2500	0	0	0	0	0	0	0
-2.0000	-0.7500	-0.2500	4.0000	1.0000	1.0000	-1.5000	-0.5000	-0.5000	0	-0.2500	-0.2500
-0.7500	-1.5000	-0.2500	1.0000	4.0000	1.0000	-0.5000	0	-0.5000	-0.2500	-1.5000	-0.7500
-0.2500	-0.2500	0	1.0000	1.0000	4.0000	-0.5000	-0.5000	-1.5000	-0.2500	-0.7500	-2.0000
0	0	0	-1.5000	-0.5000	-0.5000	4.0000	1.0000	1.5000	0	0	0
0	0	0	-0.5000	0	-0.5000	1.0000	3.0000	1.0000	0	0	0
0	0	0	-0.5000	-0.5000	-1.5000	1.5000	1.0000	4.0000	0	0	0
0	0	0	0	-0.2500	-0.2500	0	0	0	4.0000	0.7500	1.2500
0	0	0	-0.2500	-1.5000	-0.7500	0	0	0	0.7500	3.5000	1.2500
0	0	0	-0.2500	-0.7500	-2.0000	0	0	0	1.2500	1.2500	4.0000

1
2
3
0
0
0
4
4
4
3
2
1
1
2
3
0
0
0
4
4
4
3
2
1

The RHS: Global matrix

The LHS: vector

Solve 3D Equation



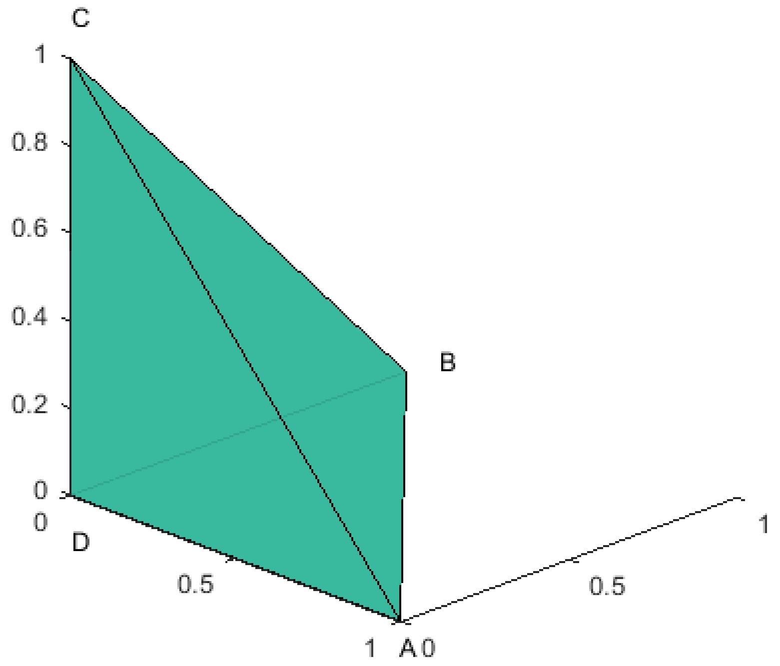
Nodal point *8

Tetrahedron *6

Inter face *6

Boundary face *12

Solve 3D Equation



lagrange interpolation

$\varphi_A = x$

$\varphi_B = y$

$\varphi_C = z$

$\varphi_D = 1-x-y-z$

Solve 3D Equation

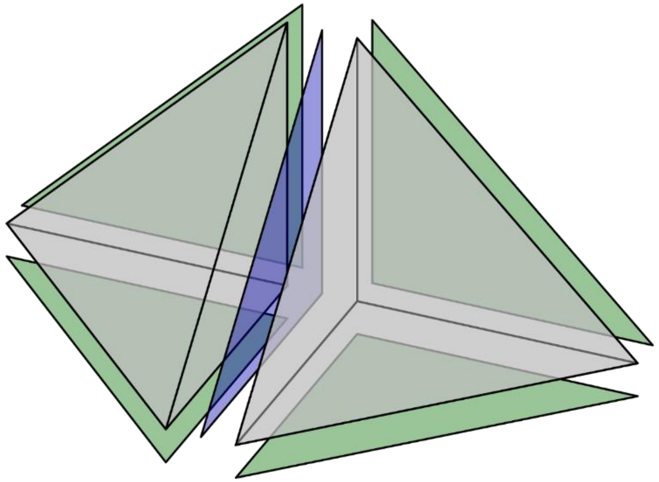
When our basic function is linear, the number of quadrature points needed for each term:

	2D	**3D**
Element term	Integration on surface	Integration of 3 dimension
Number of quadrature points	0	0
Jump term	Integration on line	Integration on face
Number of quadrature points	1	1
Penalty term	Integration on line	Integration on face
Number of quadrature points	2	3

To calculate the penalty term of 3D case, we need to figure out how the three pairs of nodal points are matched.

Solve 3D Equation

1. Find the way of mapping between the two cell and the master cell respectively
2. Find out the relation between the two cell
3. Find the way of matching
4. Get the right answer



Index	match
0	0, 1, 2
1	0, 2, 1
2	1, 0, 2
3	1, 2, 0
4	2, 0, 1
5	2, 1, 0

Solve 3D Equation

Cell1: [1, 2, 3, 4]   Cell2: [3, 1, 5, 4]

1. [1, 2, 3, 4] and [3, 1, 5, 4] are all map to [A, B, C, D]
2. The common face is [1, 3, 4] and [3, 1, 4]
3. The match way is [1, 0, 2]
4. Face [1, 3, 4] maps to [A, C, D] of the master cell
5. Face [3, 1, 4] maps to [A, B, D] of the master cell
6. Get the value of given functions at quadrature points on [A, C, D] and [A, B, D]
7. Multiply those value according to the match way

Index	**match**
0	0, 1, 2
1	0, 2, 1
2	1, 0, 2
3	1, 2, 0
4	2, 0, 1
5	2, 1, 0

Parallelization

1. Divide the global matrix into the combination of several blocks
2. Calculate the right number for each blocks in parallel
3. Combine those blocks into the global matrix
4. Use conjugate gradient method to solve the equation in parallel

Parallelization

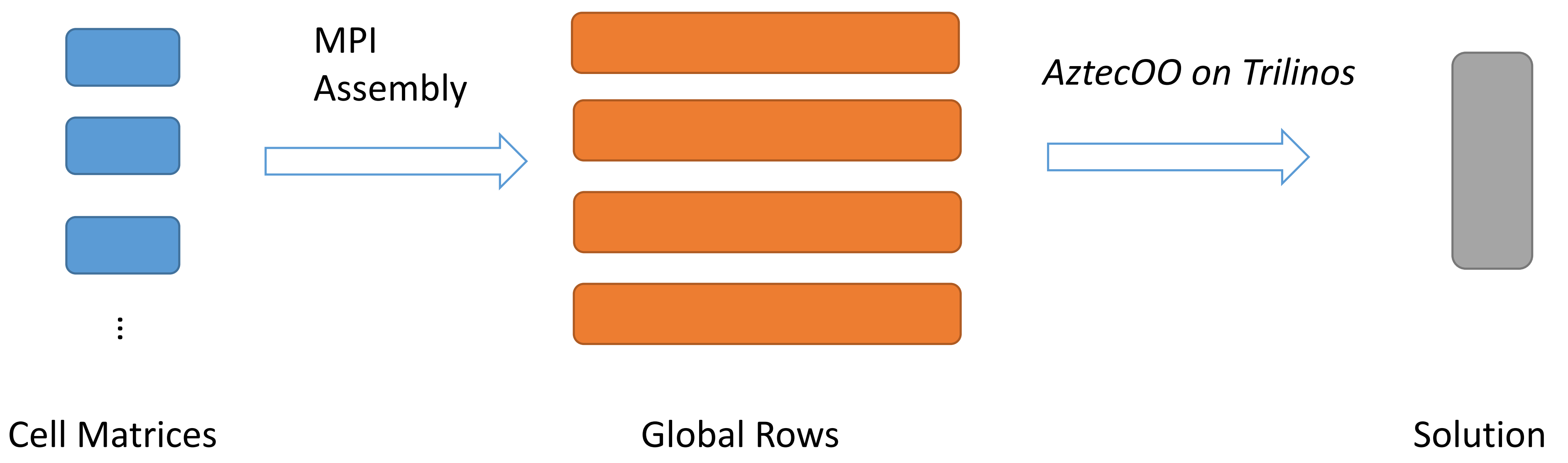
After Global Matrix Construction: Solve linear system

- C programming: *dgesv* on *LAPACK*
- LU factorization: dense matrix solver

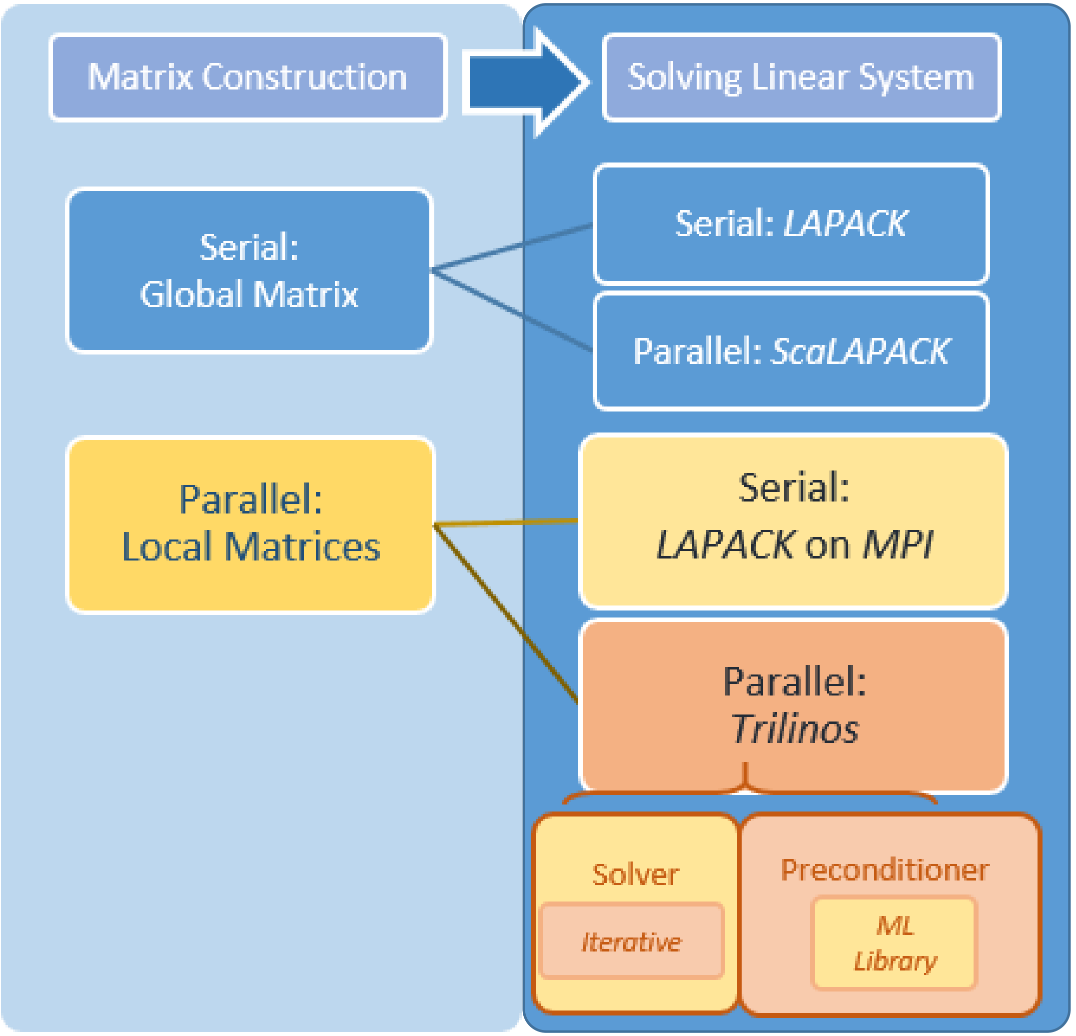
Parallelization

Parallelize the linear system solving process

- AztecOO on Trilinos
- Each processor has access to global rows



Parallelization



Future works

- Extend the partial differential equation to some time-dependent equations
- Finish the parallel code, which can be scaled on existing supercomputers.

Acknowledgement

This project is sponsored by

Oak Ridge National Laboratory

Joint Institute for Computational Sciences

University of Tennessee, Knoxville

The Chinese University of Hong Kong

Most sincere gratitude to my mentors

Dr. Ohannes Karakashian, Dr. Kwai Wong and Michael Wise

Q & A