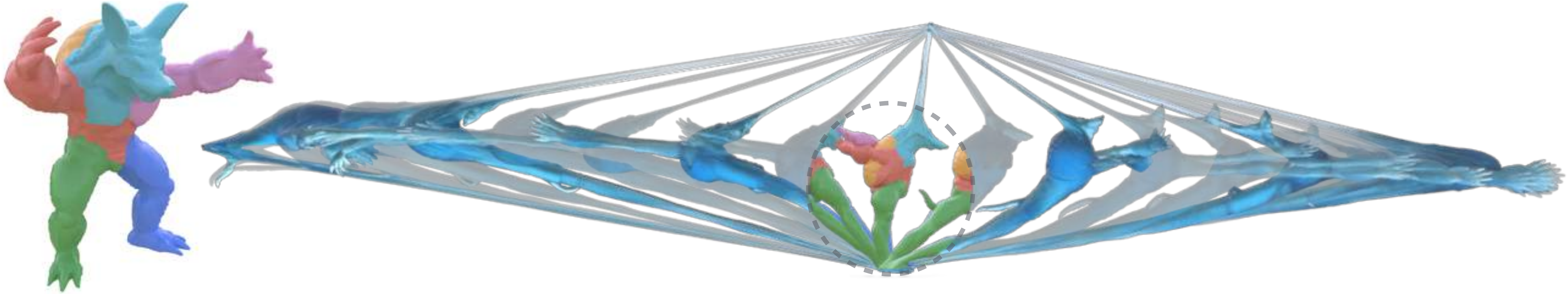


Decomposed Optimization Time Integrator for Large-Step Elastodynamics

Minchen Li[1,2], Ming Gao[1], Timothy Langlois[2], Chenfanfu Jiang[1], Danny Kaufman[2]

1. University of Pennsylvania
2. Adobe Research

Time Stepping

Optimization Time Integrator

For each time step t

Predictive position

Mass matrix

Time step Size

$$x^{t+1} = \textbf{argmin}_x E(x) = \frac{1}{2}(x - x_p)^T M(x - x_p) + h^2 W(x)$$

New node positions

Incremental potential [Ortiz and Stainier 1999]

Inertia term

Quadratic

Deformation Energy (Elasticity Potential)

Nonlinear, nonconvex! Hard!

Provide robust simulation

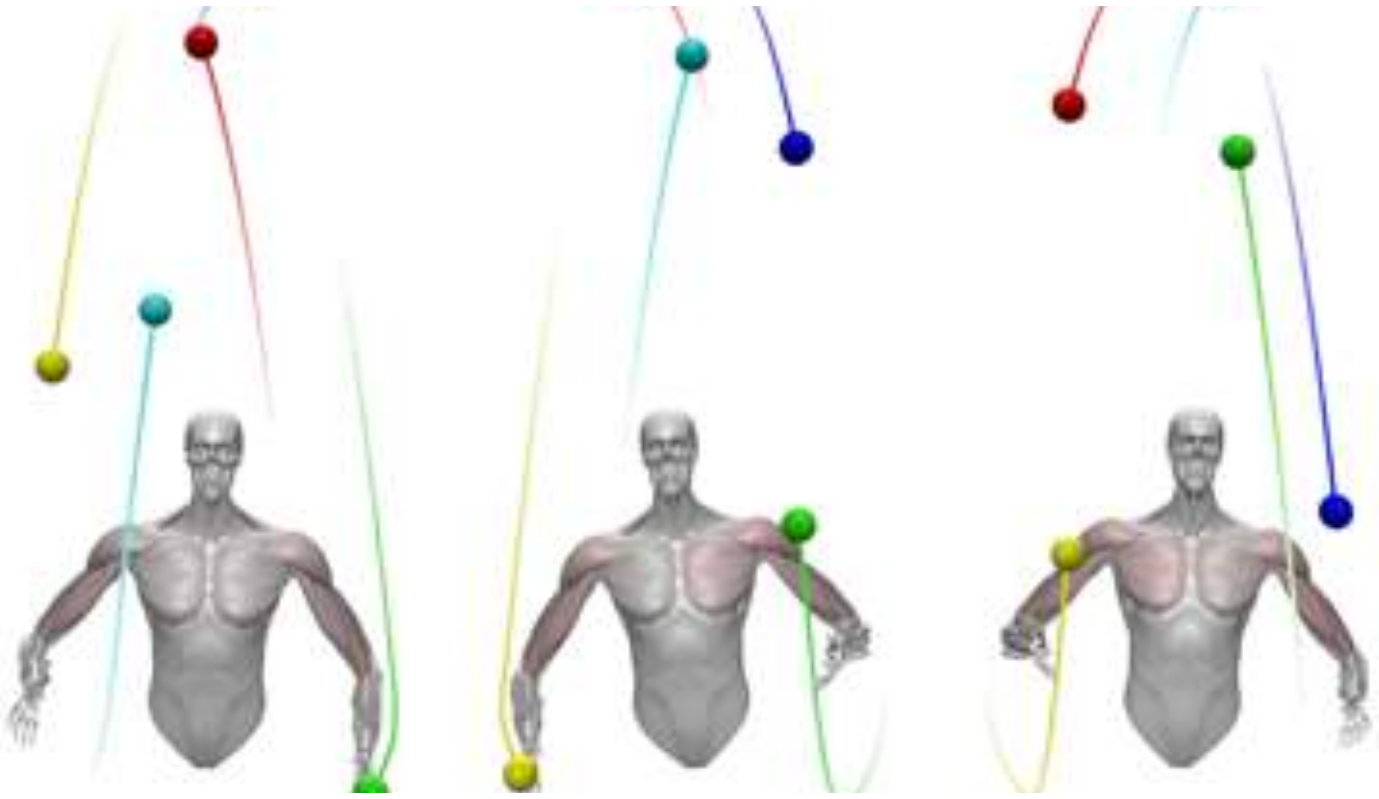
Challenging for:

- large deformation and high-speeds
- Large time step sizes h

* $x_p = x^t + hv^t + h^2 M^{-1} f$ **for implicit Euler**

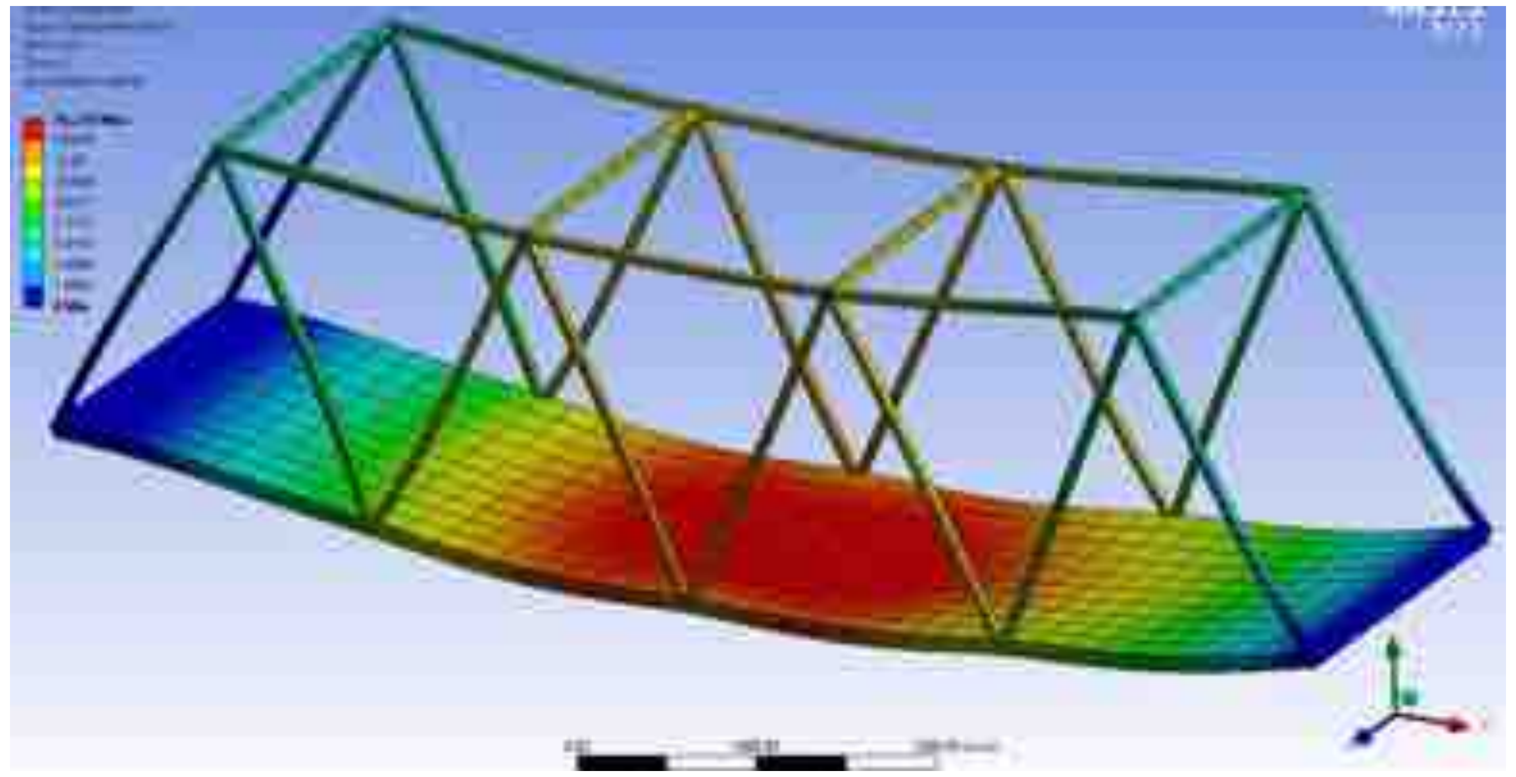
Desiderata

VFX [Smith et al. 2019]



ML [Lee et al. 2018], VR/AR, and games

Fabrication



Engineering

Robustness	**Efficiency**	**Scalability**	**Accuracy**

Line-Search Methods

1. **Precondition:** $p^i = -\boxed{P^i}^{-1}\boxed{\nabla E(x^i)}$

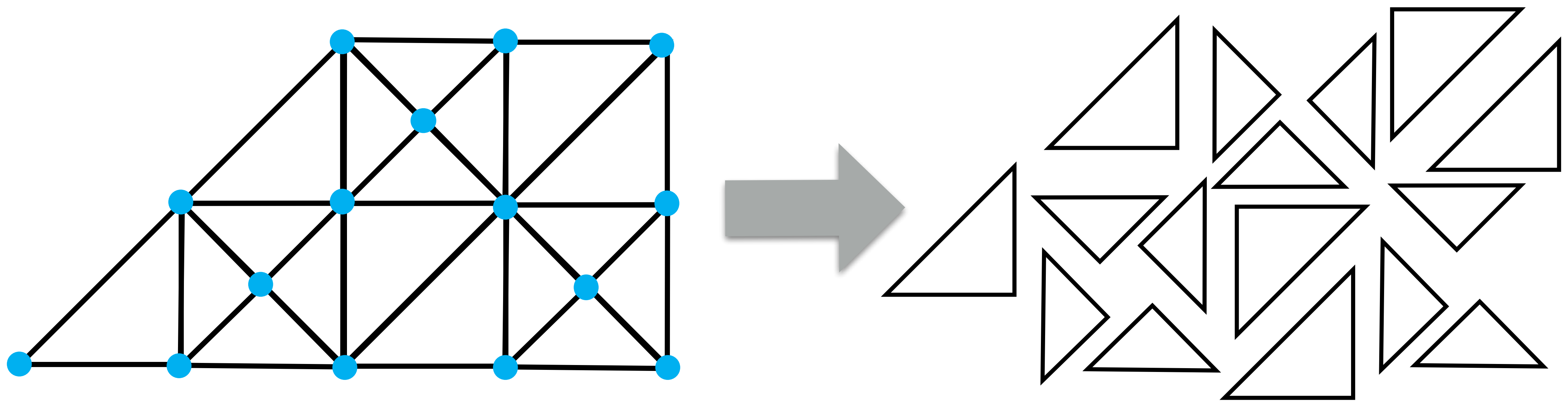
2. **Line Search:** $x^{i+1} = x^i + \alpha p^i$ **ensures** $E(x^{i+1}) \leq E(x^i)$

Methods vary in P^i :	**Efficiency**	**Scalability**	**Accuracy**
Projected Newton (PN) **[Teran et al. 2005]** $P^i = \nabla^2 E(x^i)$	✓	✗	✓
L-BFGS-H **[Brown et al. 2013]** $P^i =$ quasi-Newton initialized with $\nabla^2 E(x^i)$	✓	✗	✓
~~L-BFGS~~ L-BFGS-PD **[Liu et al. 2017]** $P^i =$ quasi-Newton initialized with $M + h^2 L$	✗	✓	

ADMM-PD [Narain et al. 2016]

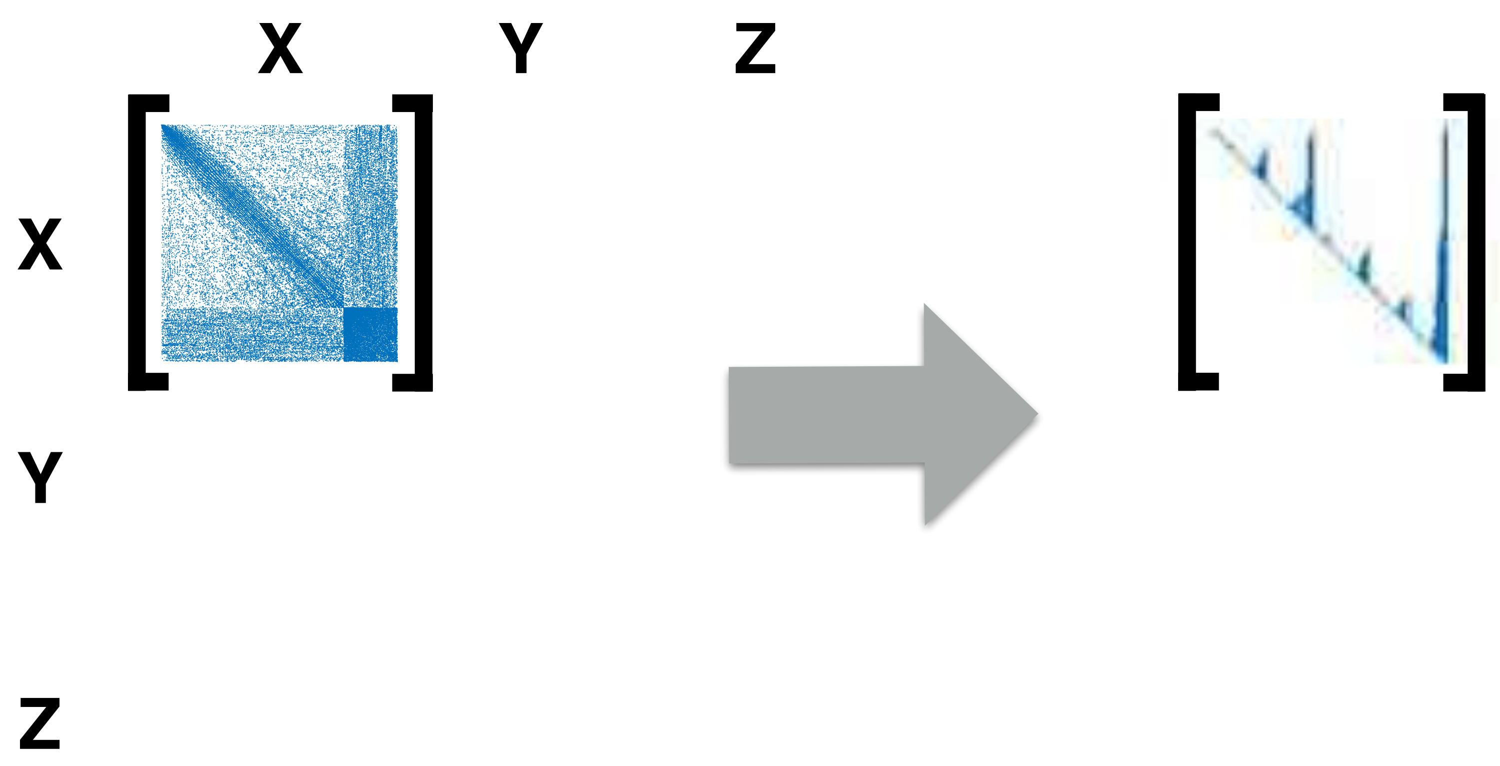
$$x^{t+1} = \text{argmin}_x E(x) = \frac{1}{2}(x - x_p)^T M(x - x_p) + h^2 W(x)$$

1. **Elasticity** solve on element soup in parallel



2. **Global solve** with $(M + h^2 L)^{-1}$

$M + h^2 L =$



Not convergent!

Efficiency	Scalability	Accuracy
✓	✓	✗

Feature Table

	Efficiency	Scalability	Accuracy
LBFGS-PD	✗	✓	✓
LBFGS-H	✓	✗	✓
PN	✓	✗	✓
ADMM-PD	✓	✓	✗
DOT	✓	✓	✓

100K tetrahedra,

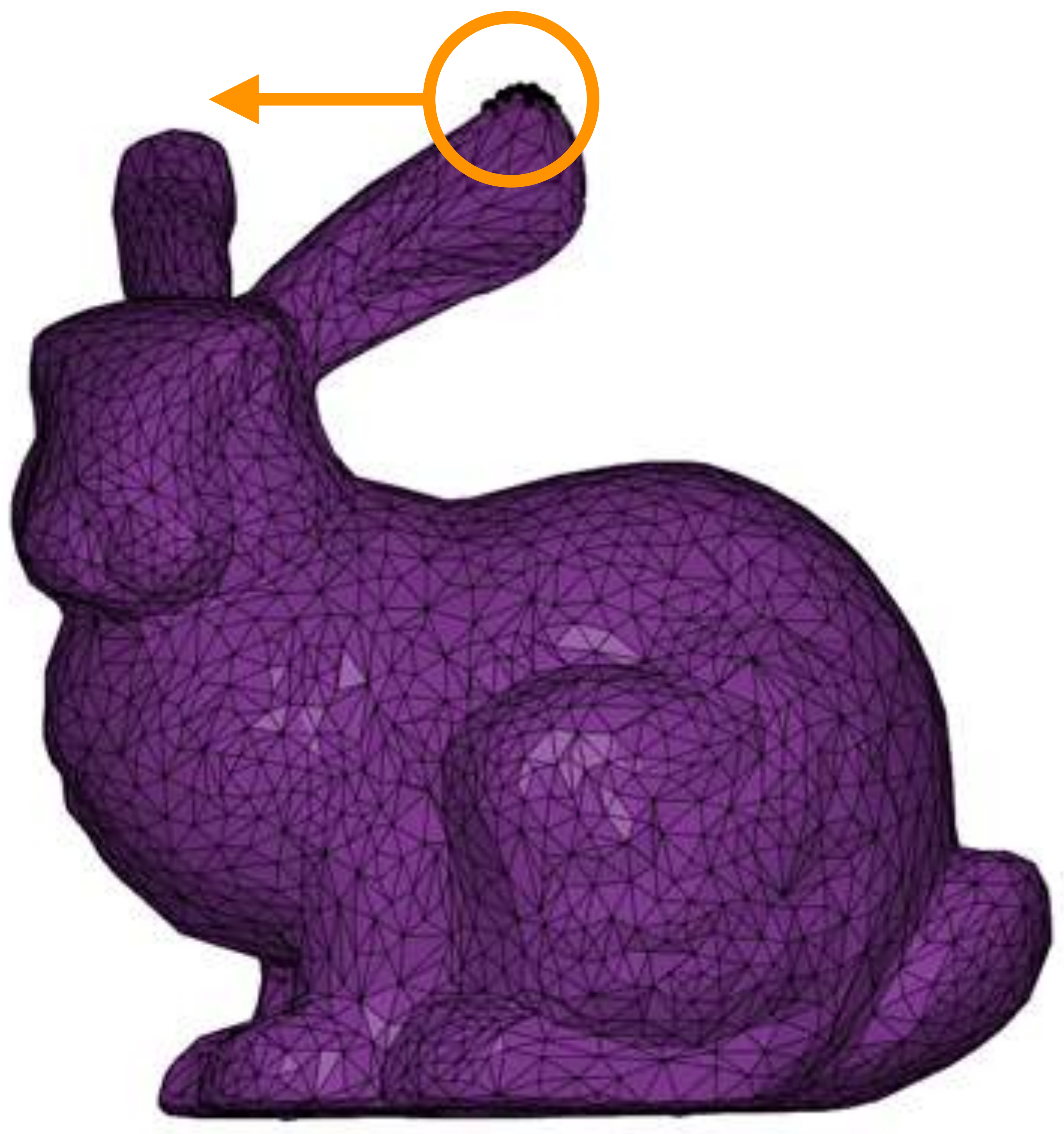
Time step size: 10ms,

Converged to 10^{-5}CN

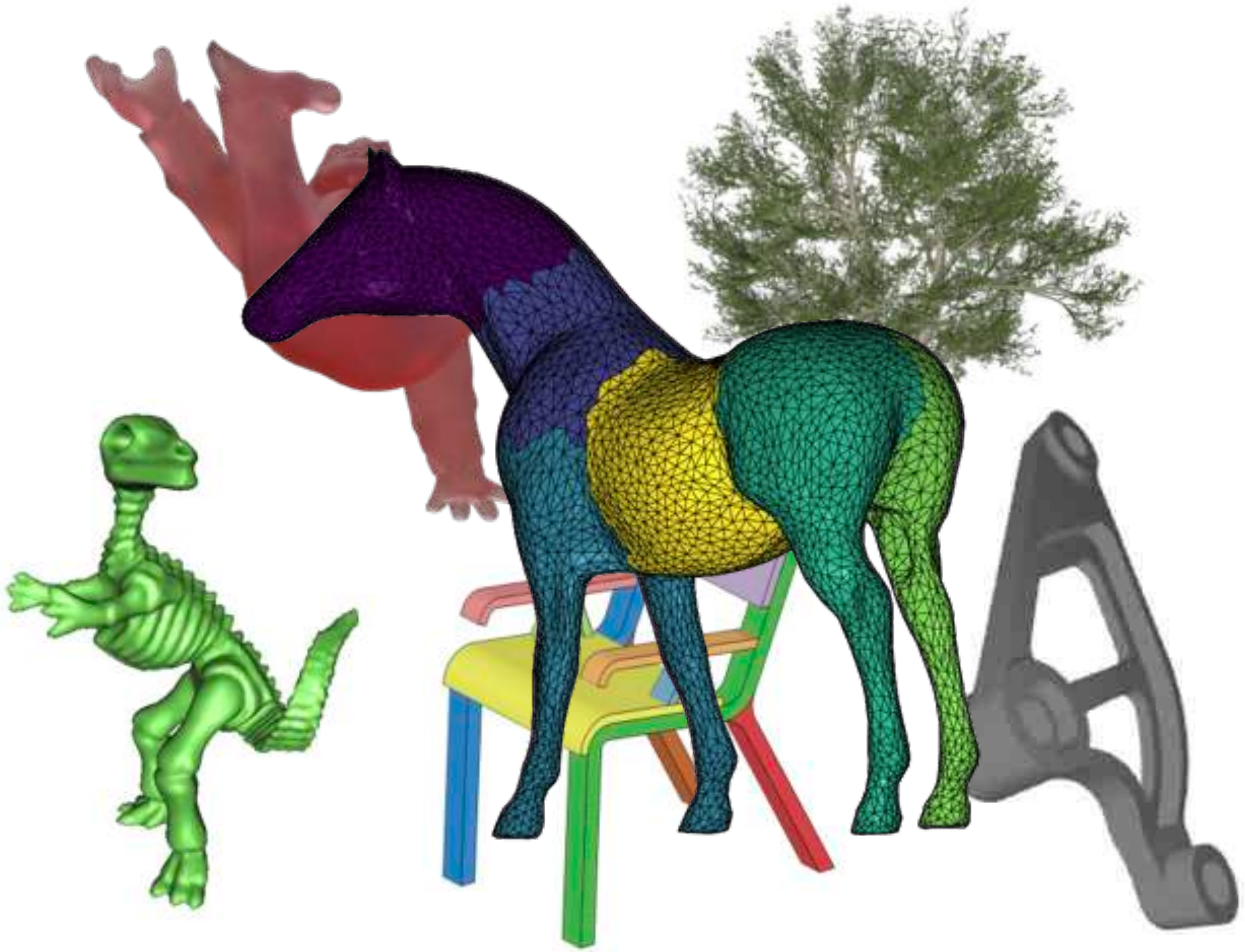
1.9 sec/frame,

Observations

Deformations are local

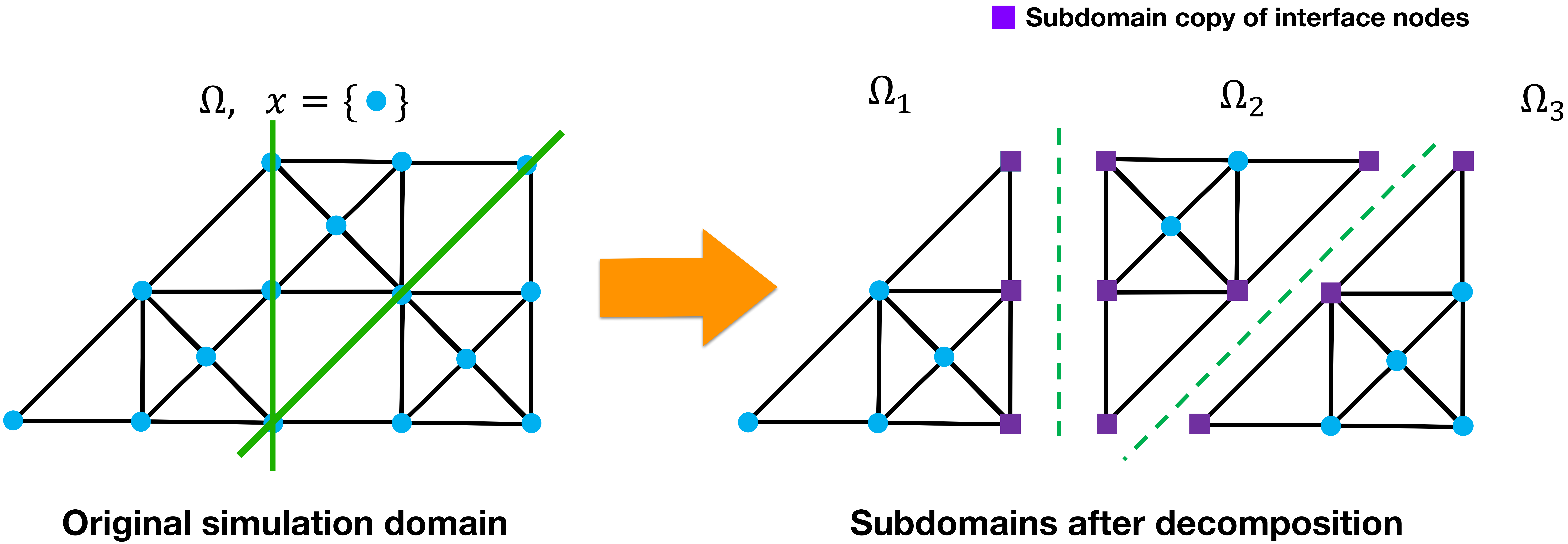


Articulated Structure



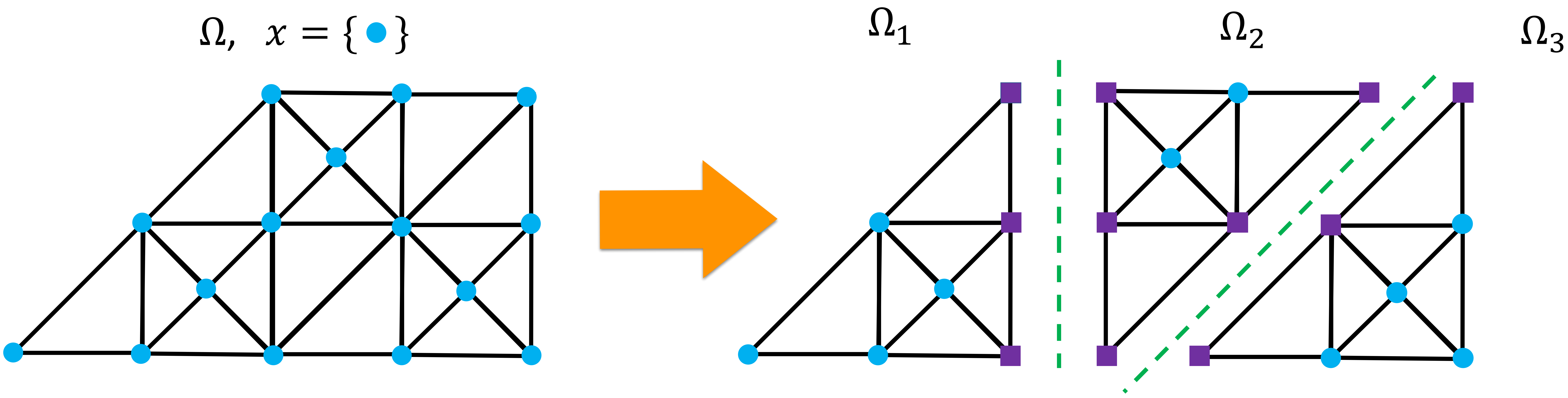
Domain Decomposition

Domain Decomposition



Domain Decomposition

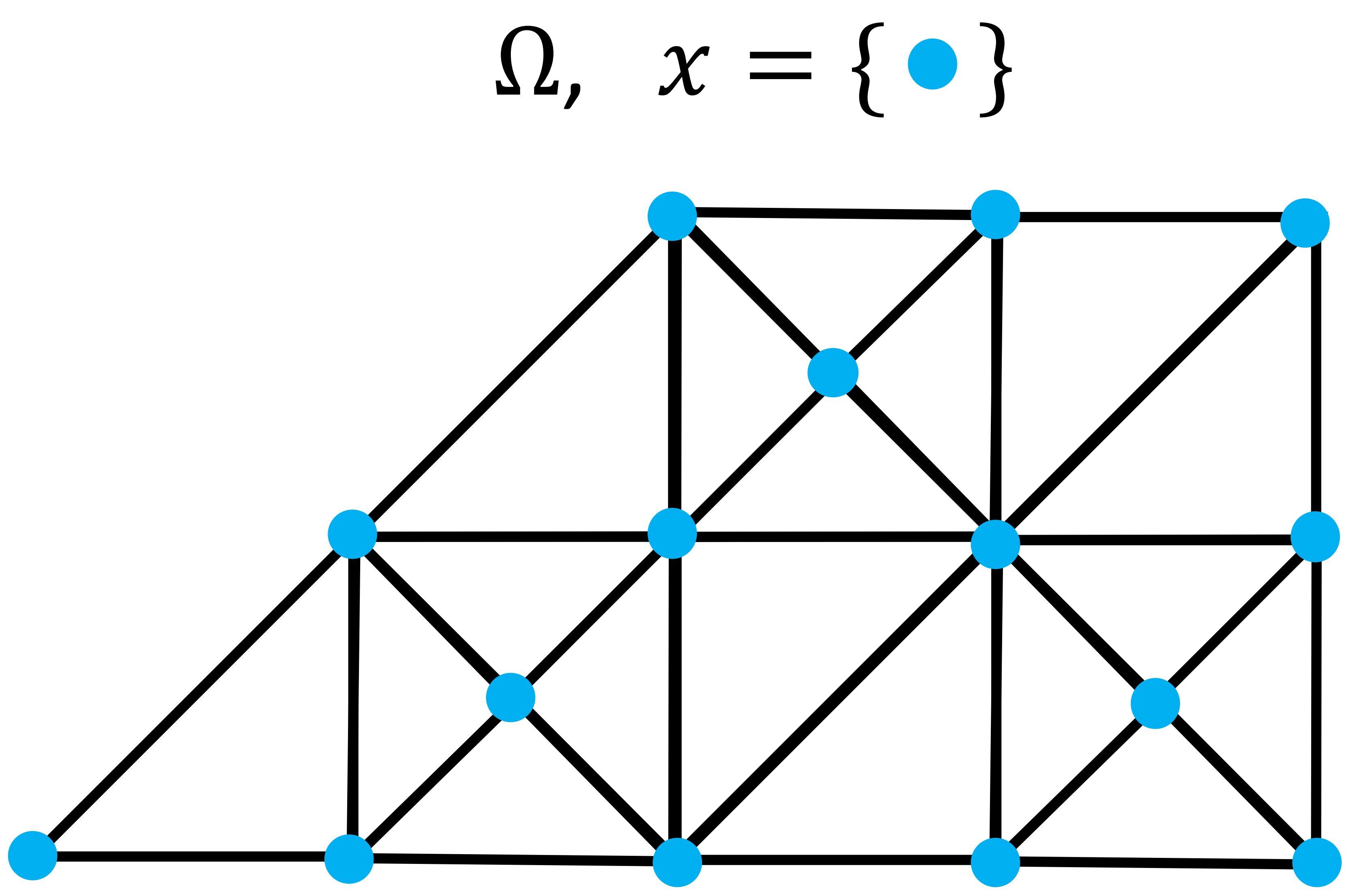
- Domain decomposition preconditions iterative linear solvers
- Extensions to nonlinear systems with slow convergence



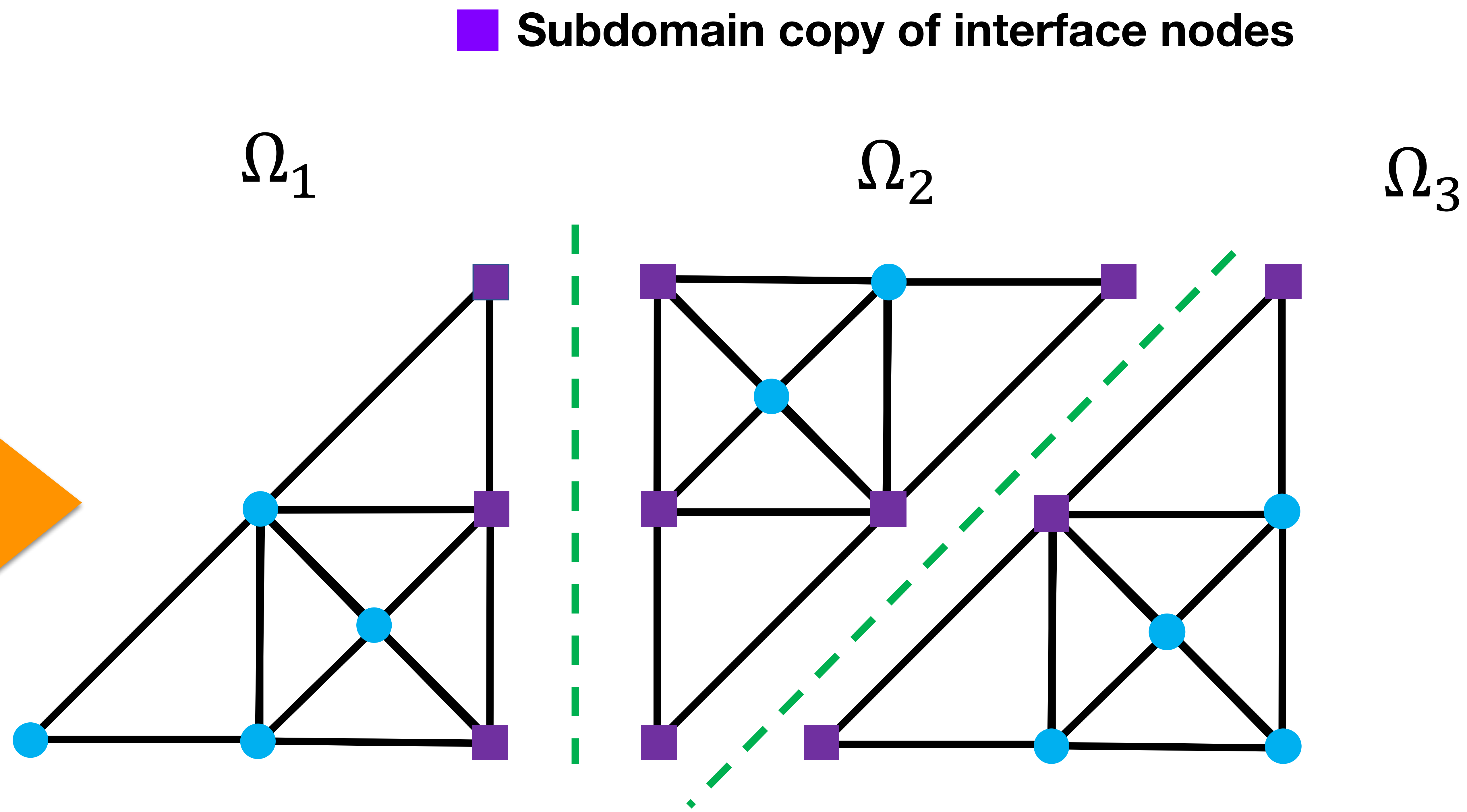
Original simulation domain **Subdomains after decomposition**

DOT Algorithm

Domain Decomposition

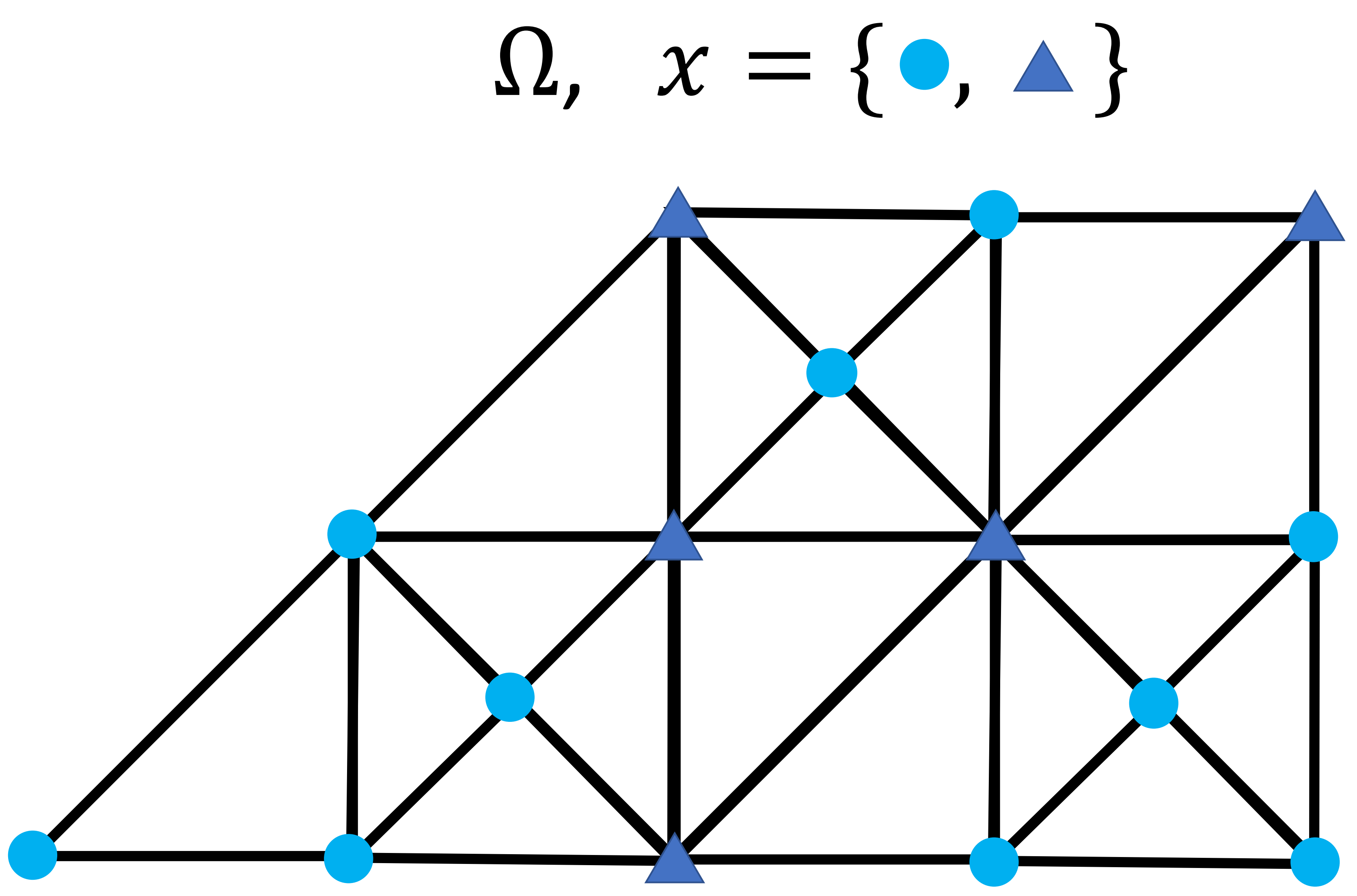


Original simulation domain

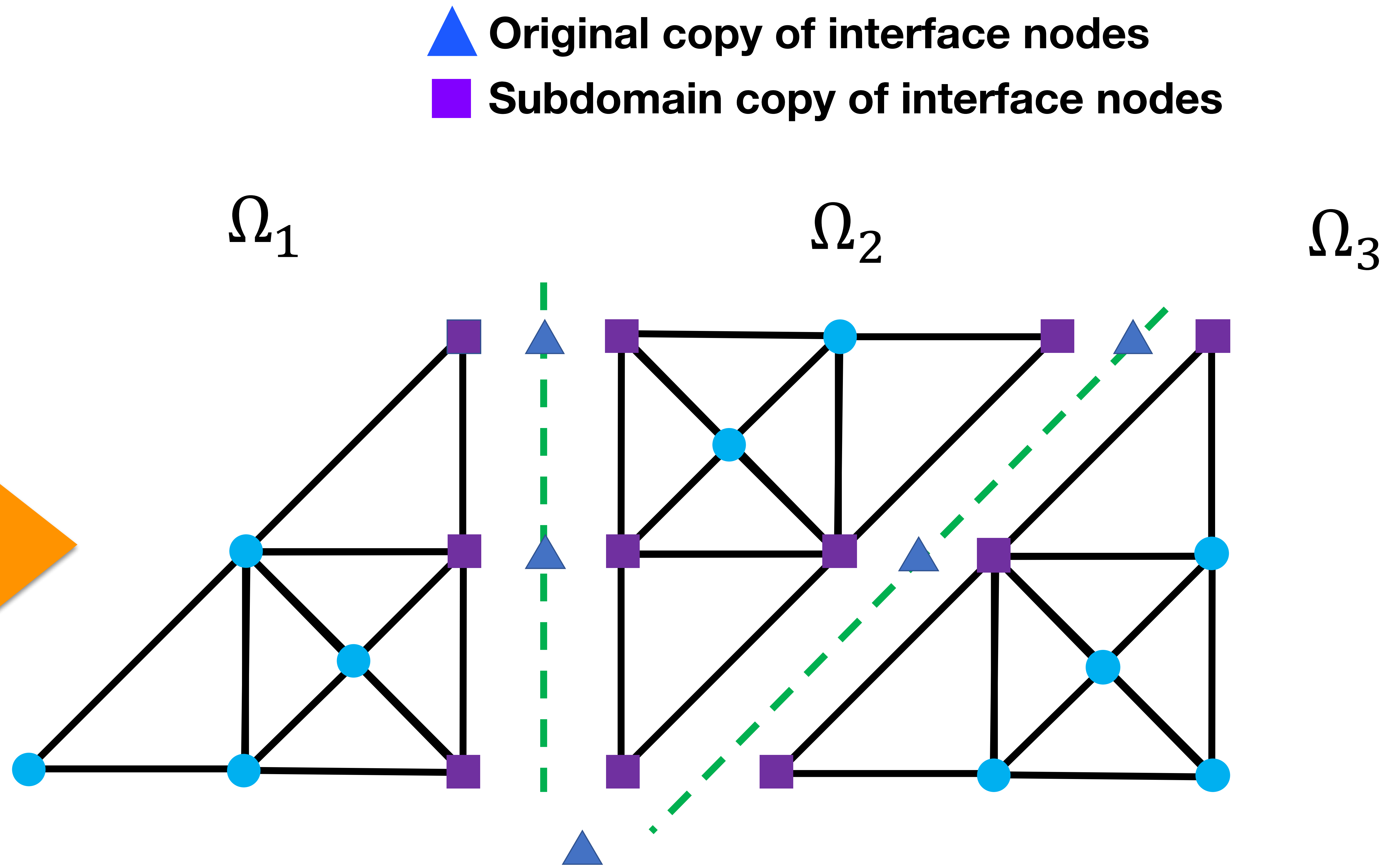


Subdomains after decomposition

Domain Decomposition

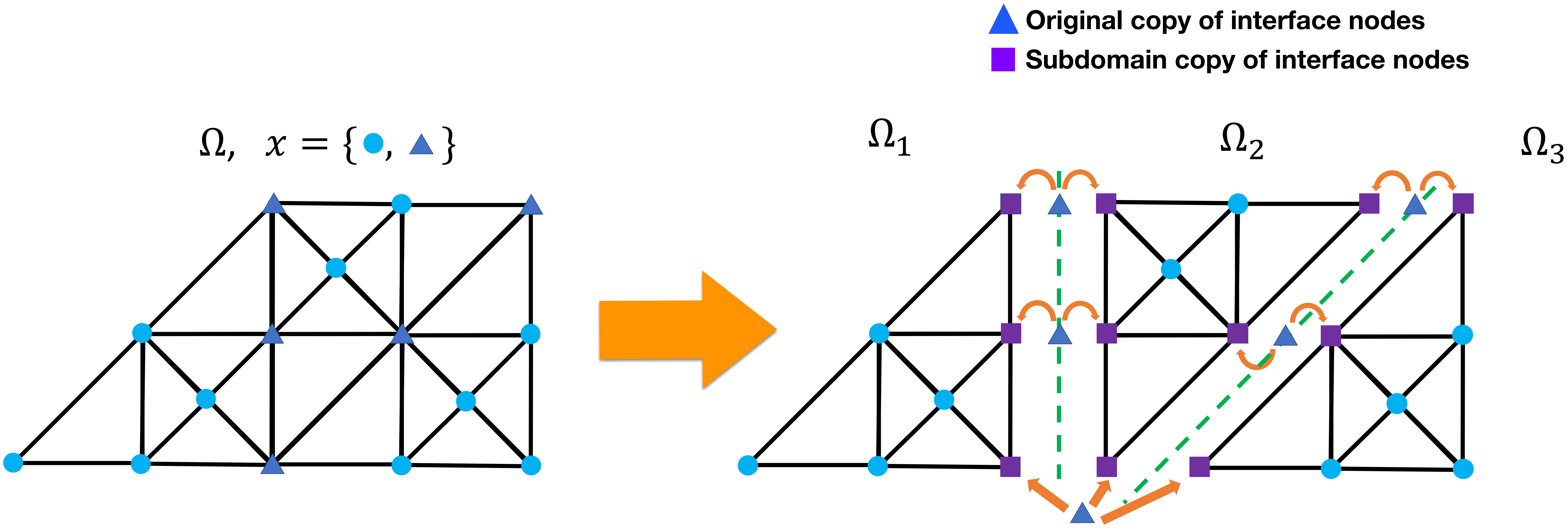


Original simulation domain



Subdomains after decomposition

Domain Decomposition



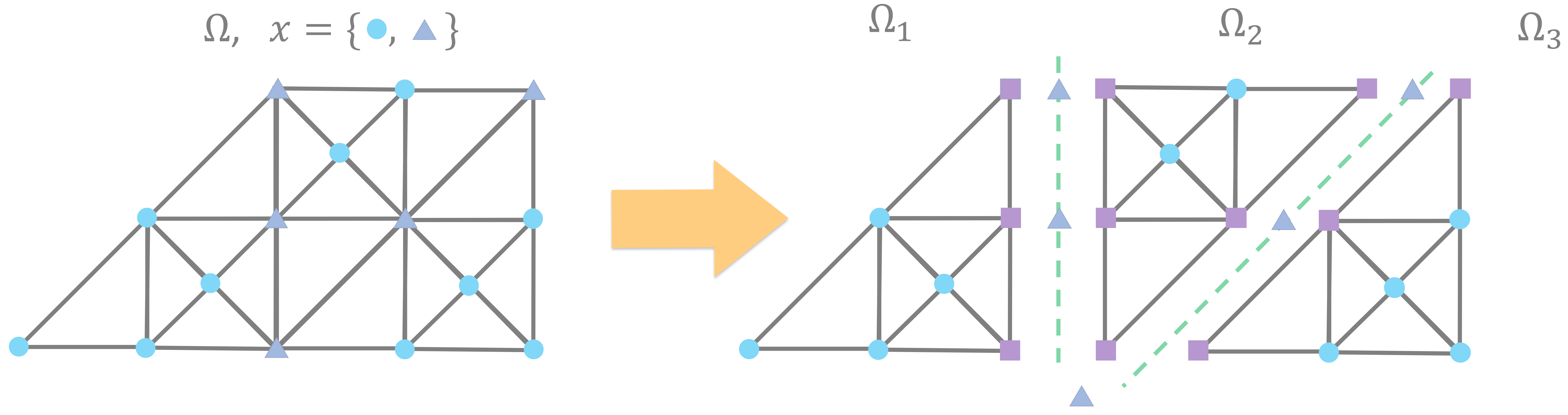
Original simulation domain **Subdomains after decomposition**

Decomposed Penalty Lagrangian

$$\min \Sigma_{\Omega_j} E_j(\bullet, \blacksquare) \quad s.t. \quad \blacksquare = \blacktriangle$$

$$L(\bullet, \blacksquare, \blacktriangle) = \Sigma_{\Omega_i}\left(E_j(\bullet, \blacksquare) + \frac{1}{2}(\blacksquare - \blacktriangle)^T K_j (\blacksquare - \blacktriangle)\right)$$

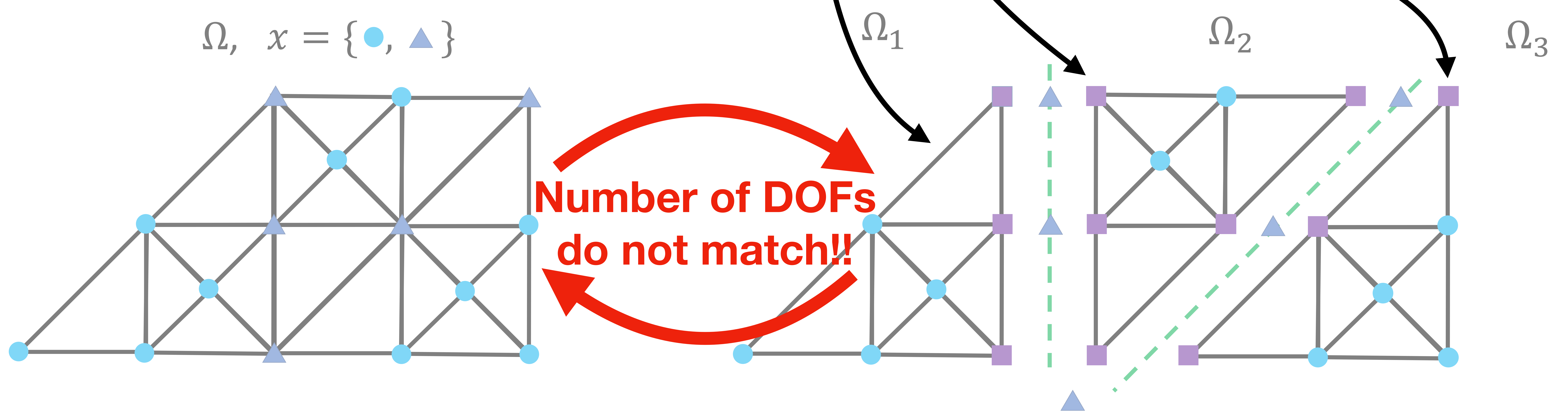
ill-conditioning!!



Decomposed Initializer

For inner initializer
Of LBFGS!

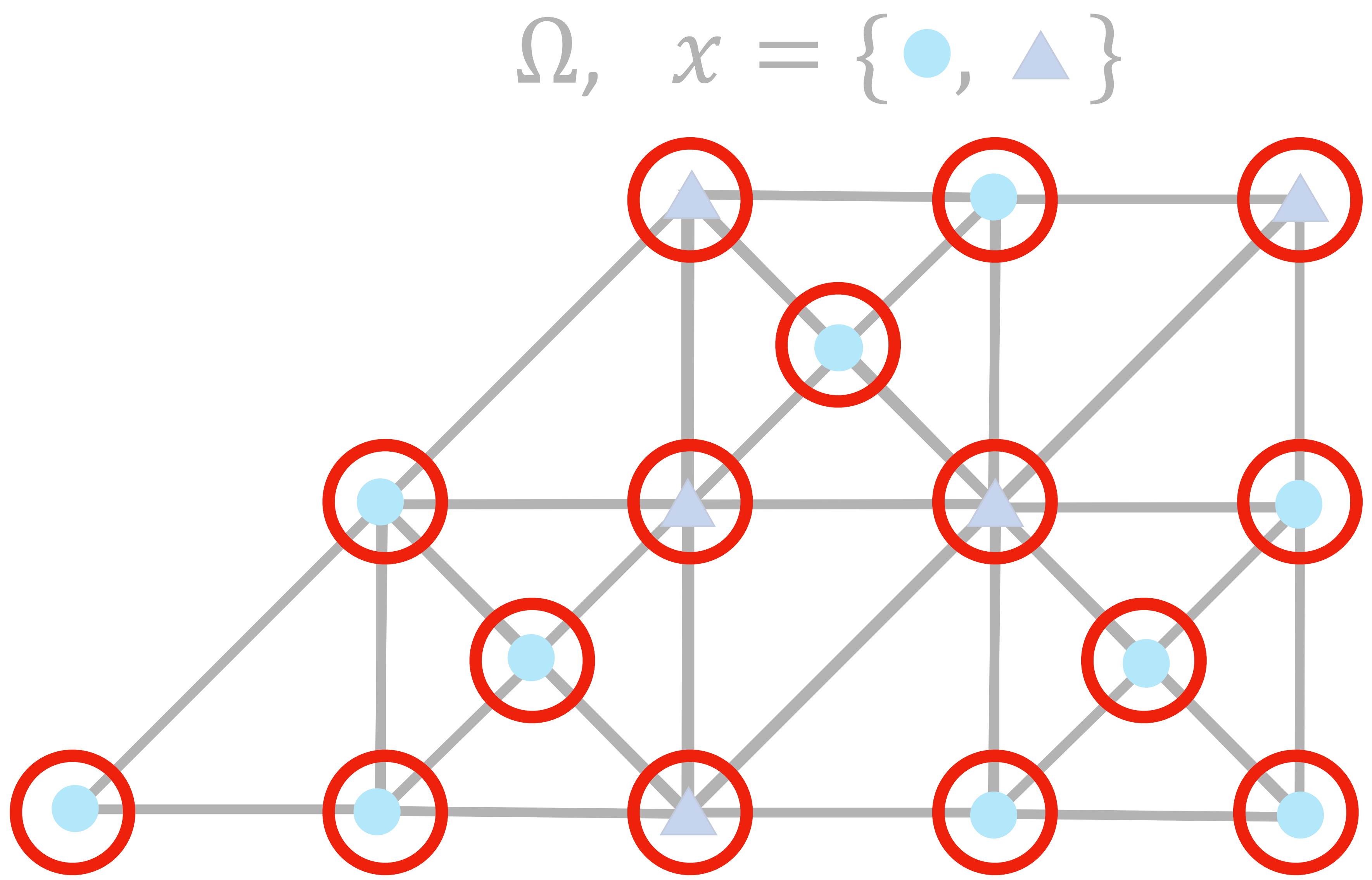
The penalty Hessian: $\frac{\partial^2 L}{\partial\{\,\bullet\,,\,\blacksquare\,\}^2} = \begin{bmatrix} H_1 & & \\ & H_2 & \\ & & H_3 \end{bmatrix}$

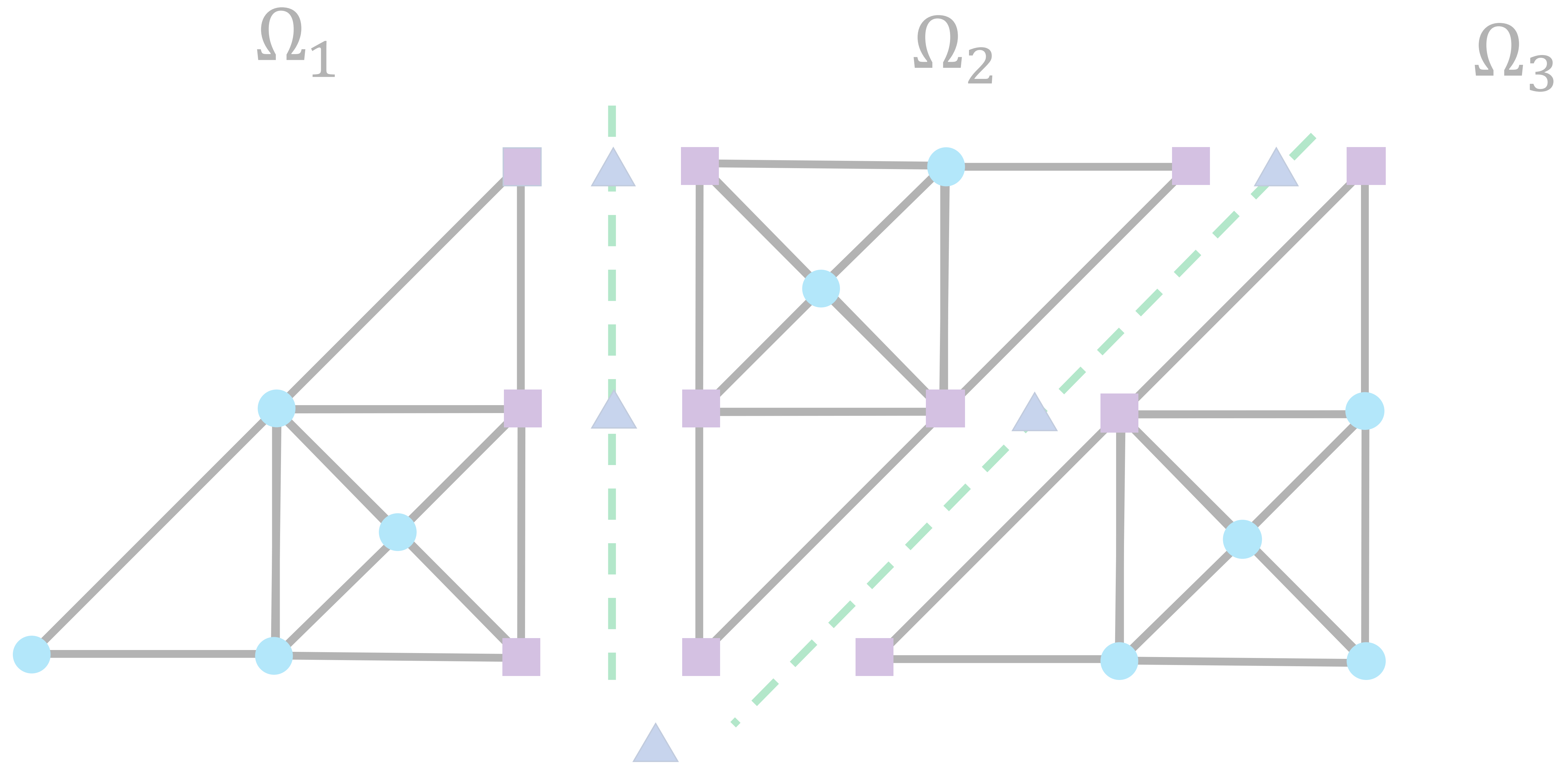


Decomposed Initializer

Vector defined on original domain

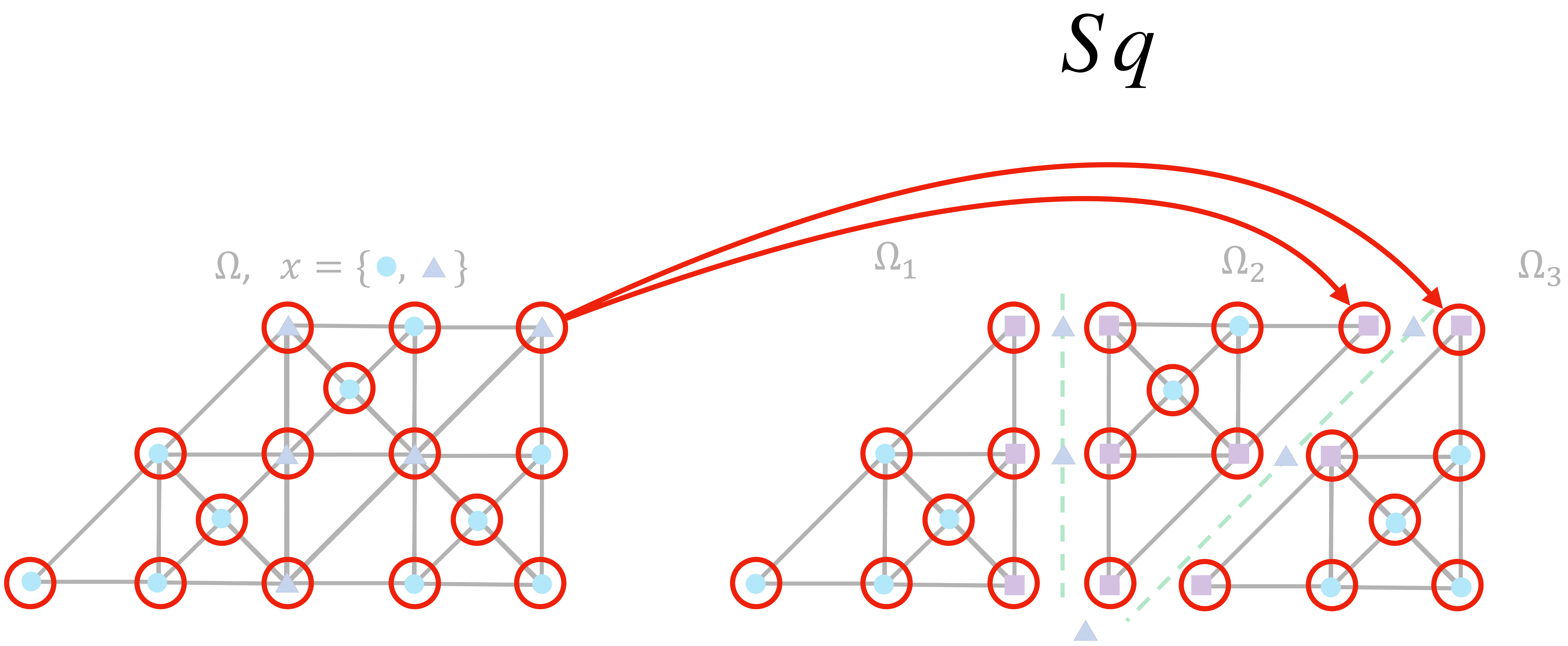
q





Decomposed Initializer

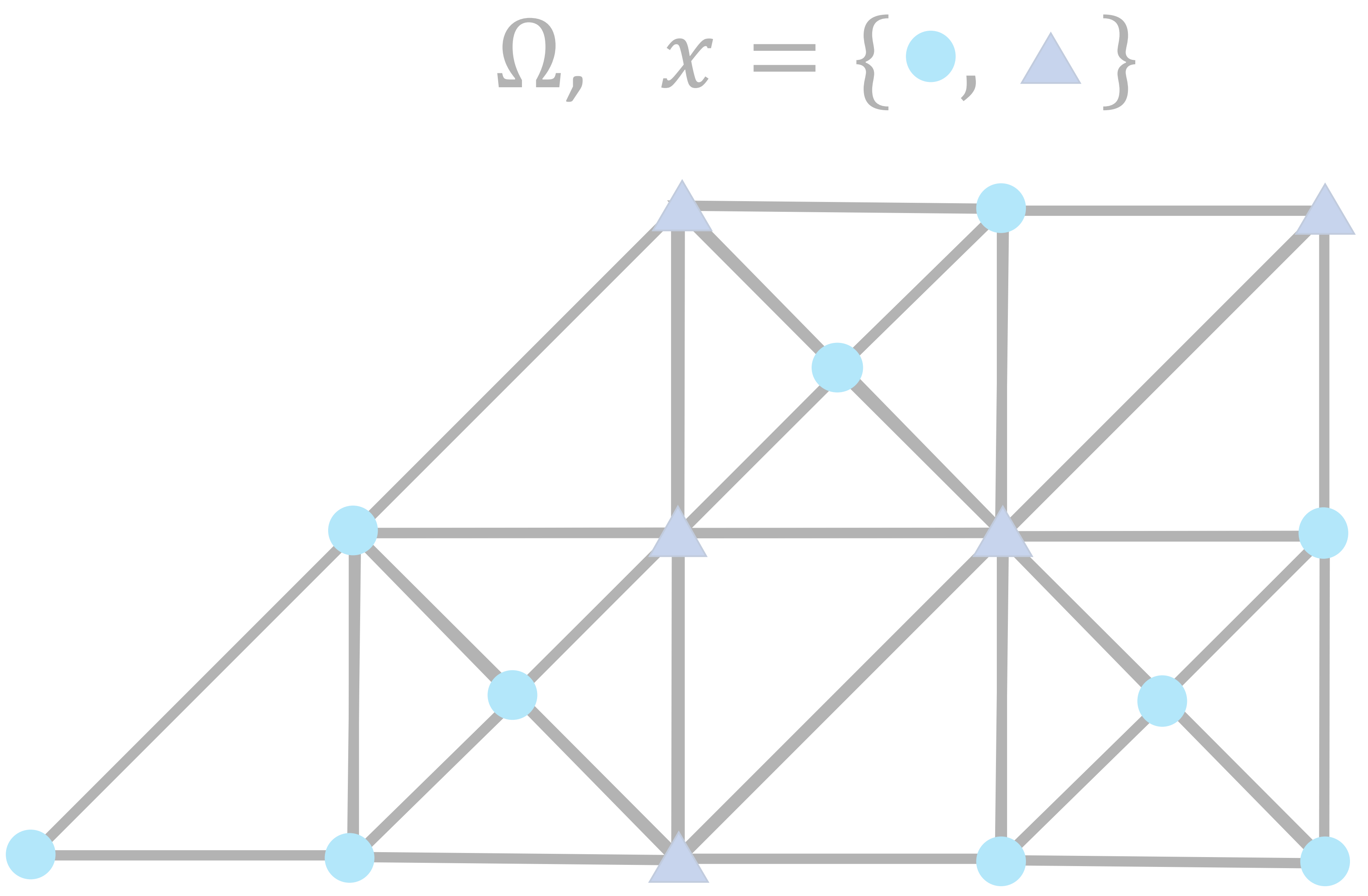
Vector from original domain to subdomains

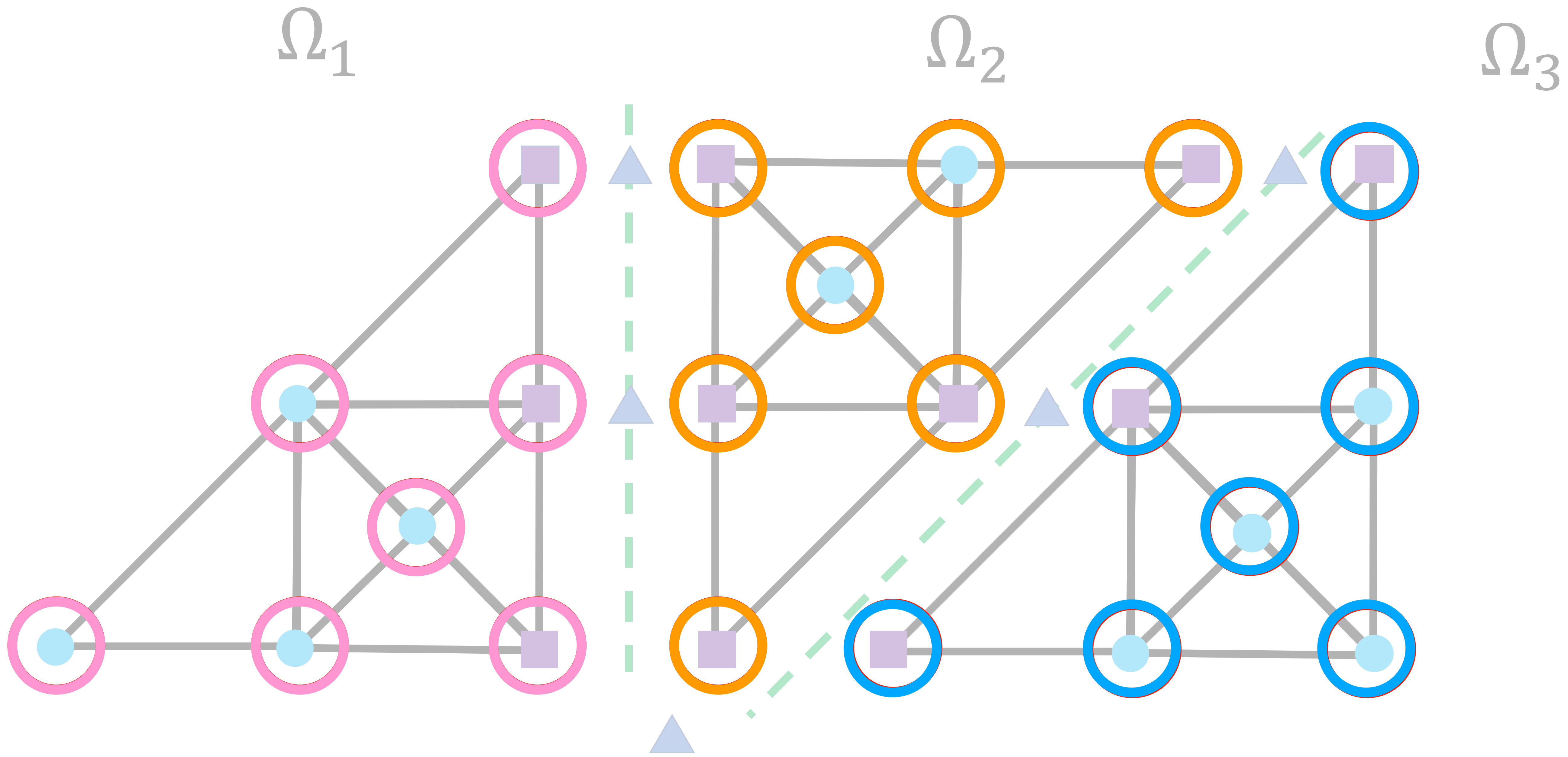


Decomposed Initializer

Independent per domain back solves

$$\begin{bmatrix} H_1^{-1} & & \\ & H_2^{-1} & \\ & & H_3^{-1} \end{bmatrix} S\,q$$

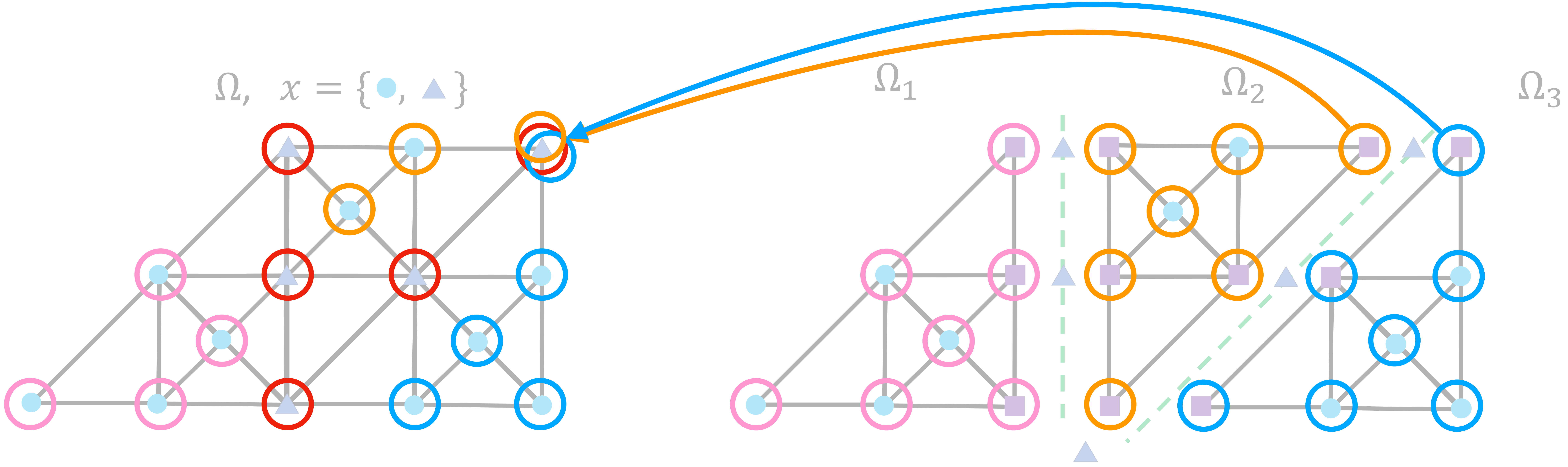




Decomposed Initializer

Vector back to original domain

$$r = \boxed{BS^T \begin{bmatrix} H_1^{-1} & & \\ & H_2^{-1} & \\ & & H_3^{-1} \end{bmatrix} S}\, q$$



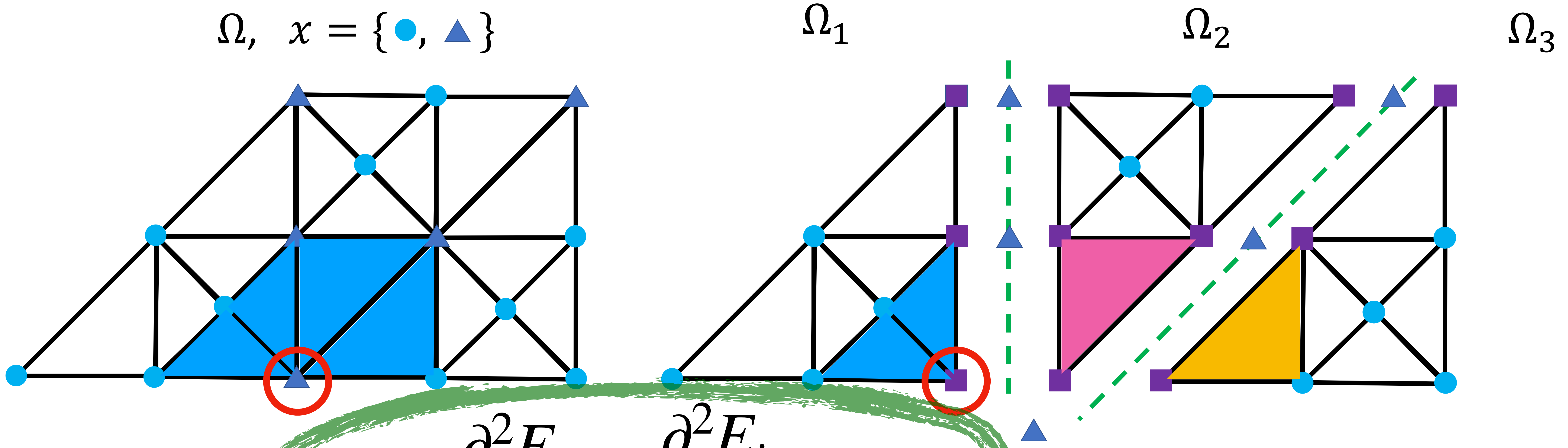
Penalty Stiffness

$$L(\bullet, \blacksquare, \blacktriangle) = \Sigma_{\Omega_i}\left(E_j(\bullet, \blacksquare) + \frac{1}{2}(\blacksquare - \blacktriangle)^T K_j (\blacksquare - \blacktriangle)\right)$$

?

Penalty Stiffness

Subdomain Hessian: $$H_j = \frac{\partial^2 L}{\partial\{\bullet_j,\ \blacksquare_j\}^2} = \begin{bmatrix} \dfrac{\partial^2 E_j}{\partial \bullet_j^2} & \dfrac{\partial^2 E_j}{\partial \bullet_j \partial \blacksquare_j} \\[2ex] \dfrac{\partial^2 E_j}{\partial \blacksquare_j \partial \bullet_j} & \dfrac{\partial^2 E_j}{\partial \blacksquare_j^2} + K_j \end{bmatrix}$$



Use $\dfrac{\partial^2 E}{\partial \blacktriangle^2} \neq \dfrac{\partial^2 E_j}{\partial \blacksquare^2}$ for K_j

DOT Pseudo-code

While $||\nabla E(x^i)||_2 \geq \epsilon_{CN}$ // gradient residual convergence check [Zhu et al. 2018]

$q \leftarrow$ lowRankUpdate$(-\nabla E(x^i))$ // 1st quasi-Newton update

$(q_1, q_2, \ldots, q_s) \leftarrow$ separate(q) // Separate full DoFs to subdomains

$r_j \leftarrow$ backsolve$(q_j), \forall j \in [1,s]$ // Back-solve subdomains in parallel

$r \leftarrow$ merge$(r_1, r_2, \ldots, r_s)$ // Merge subdomain to full coordinates

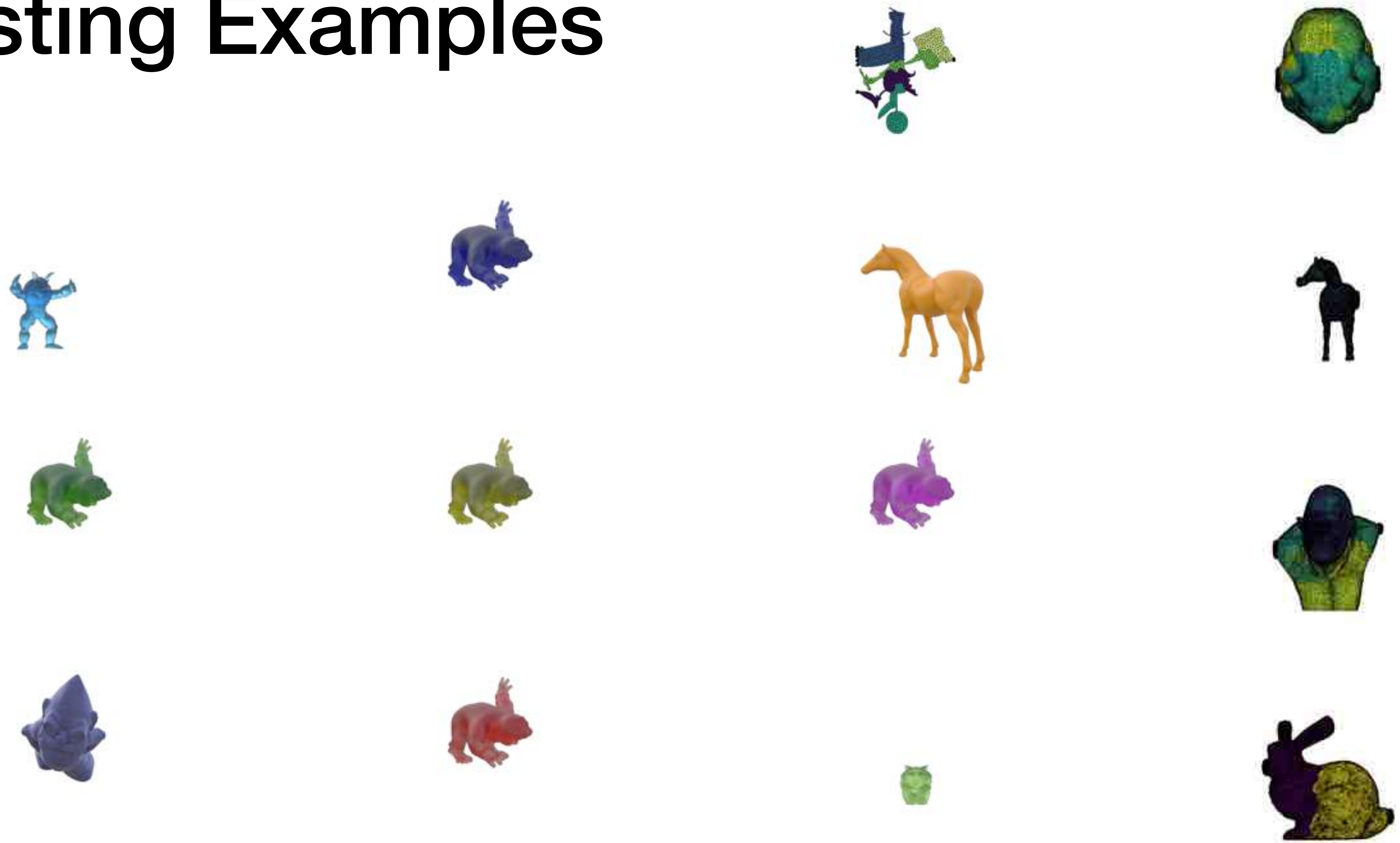
$p \leftarrow$ lowRankUpdate(r) // 2nd quasi-Newton update

$x^{i+1} \leftarrow x^i + \alpha p$ // Line-search and update

Decomposed Initialier

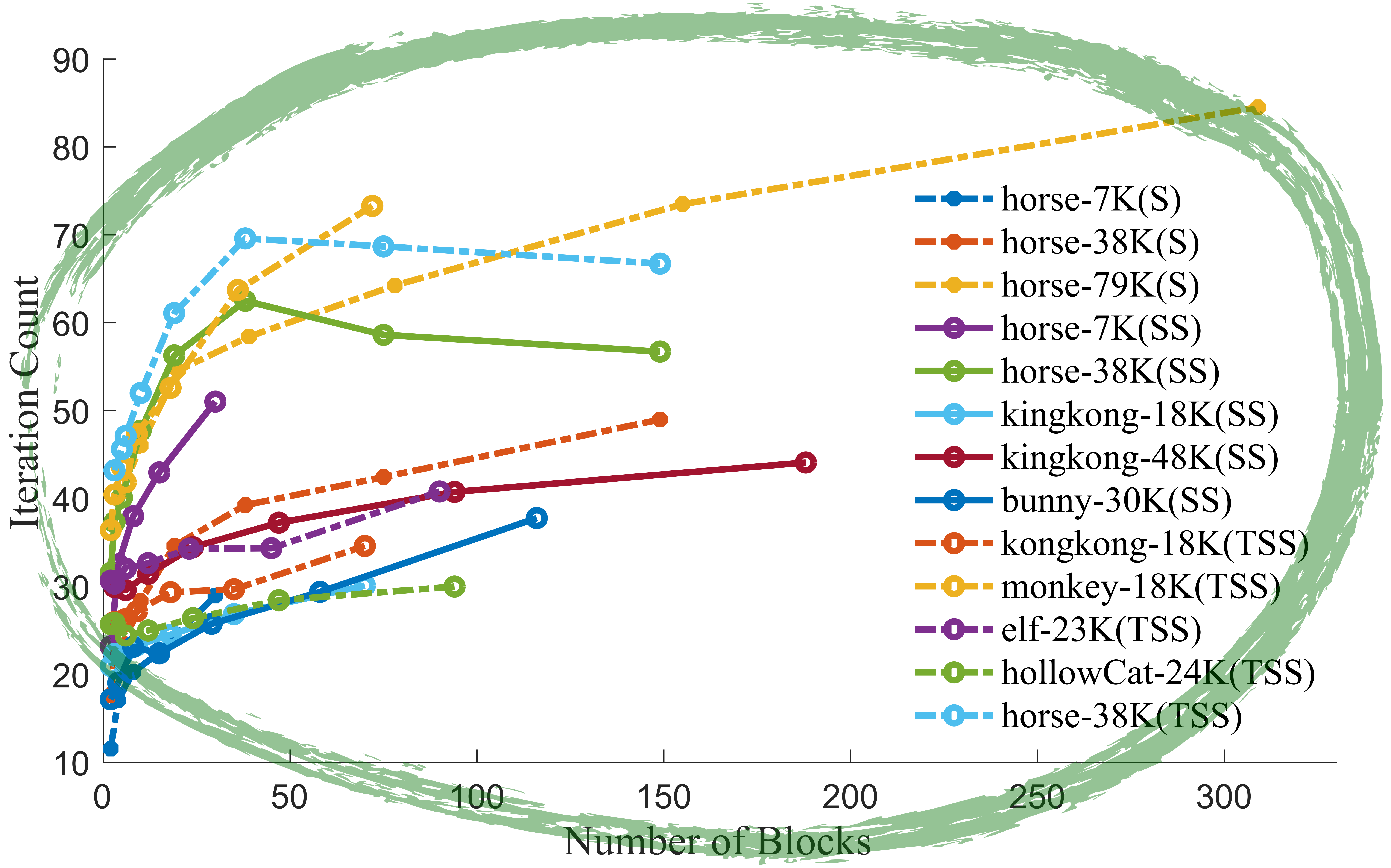
Experiments and Results

Testing Examples



DOT Iteration Growth with Subdomain Count

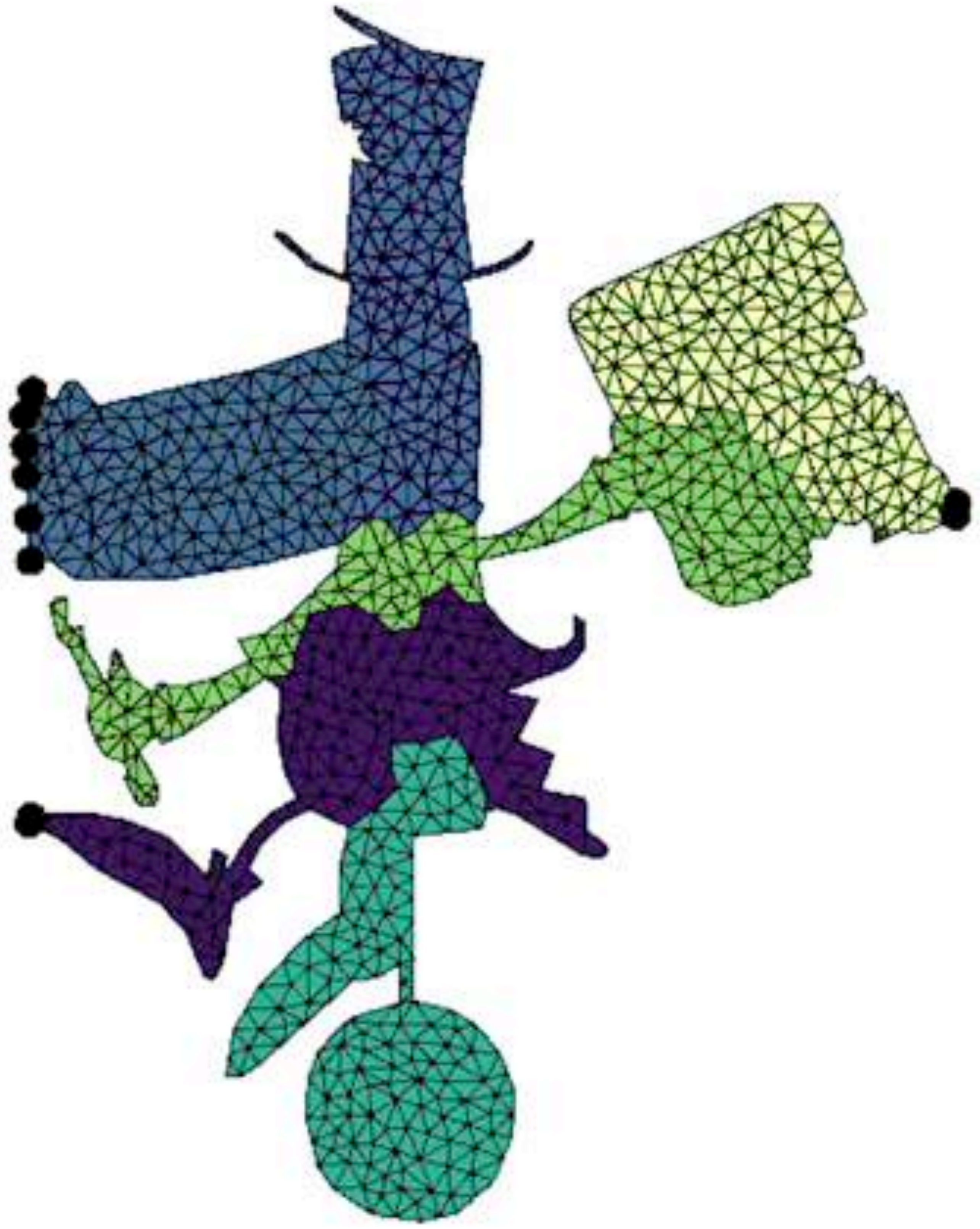
Decompose meshes with METIS **[Karypis and Kumar 2009]**



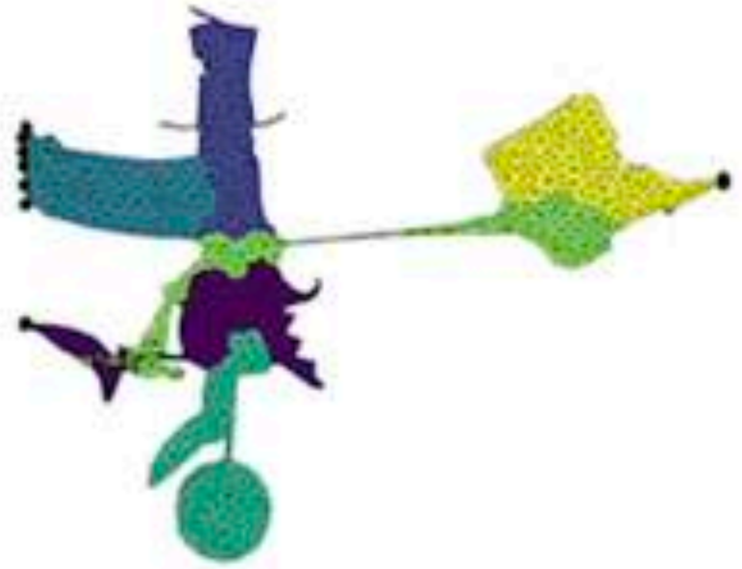
Sub-linear growth!

DOT Iteration Process

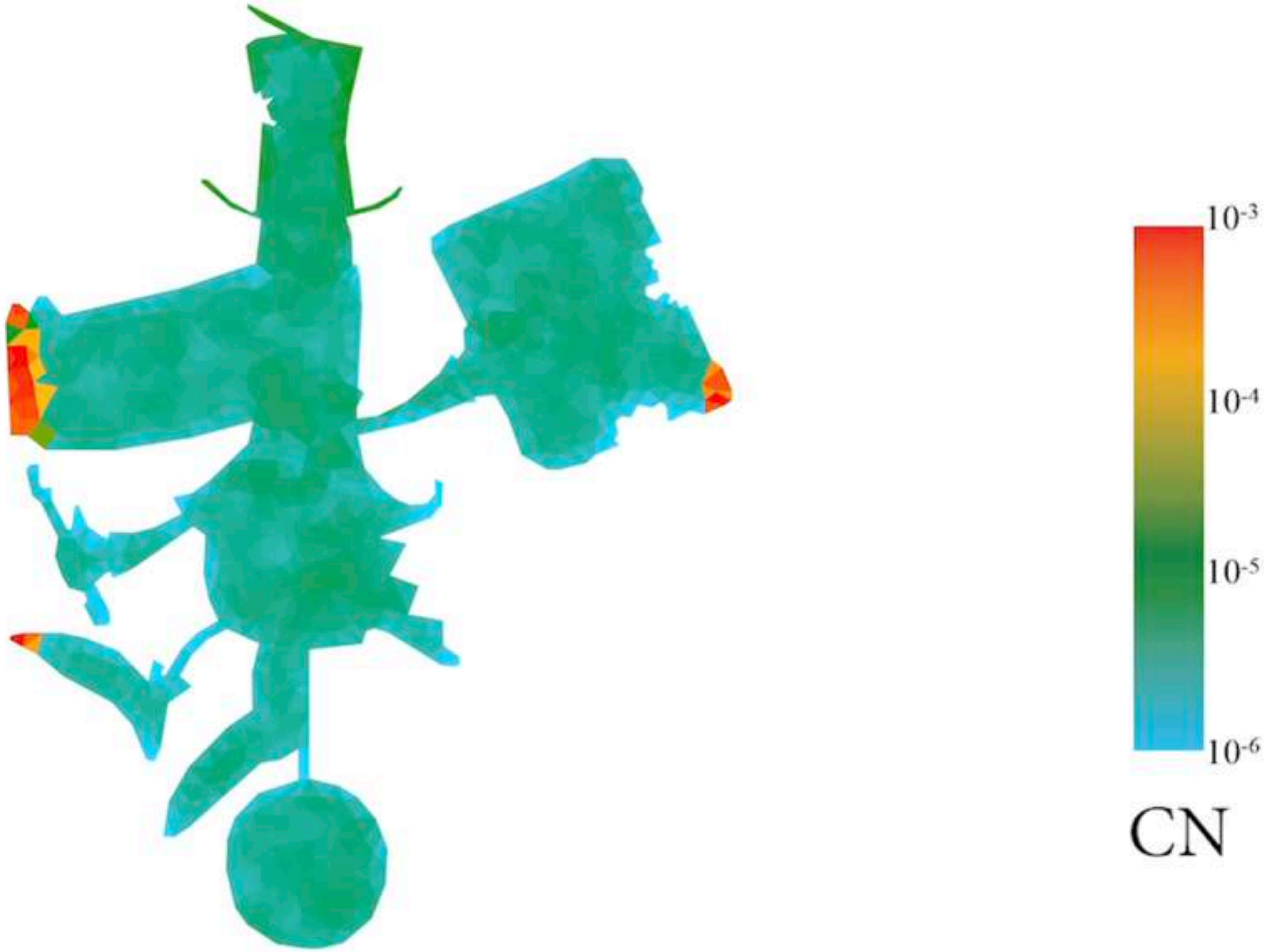
A Visualization of DOT's decomposition:



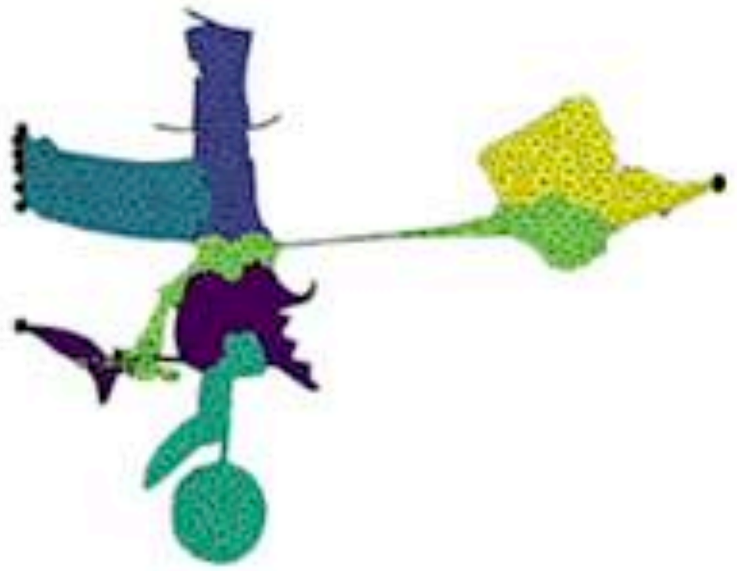
DOT Iteration Process



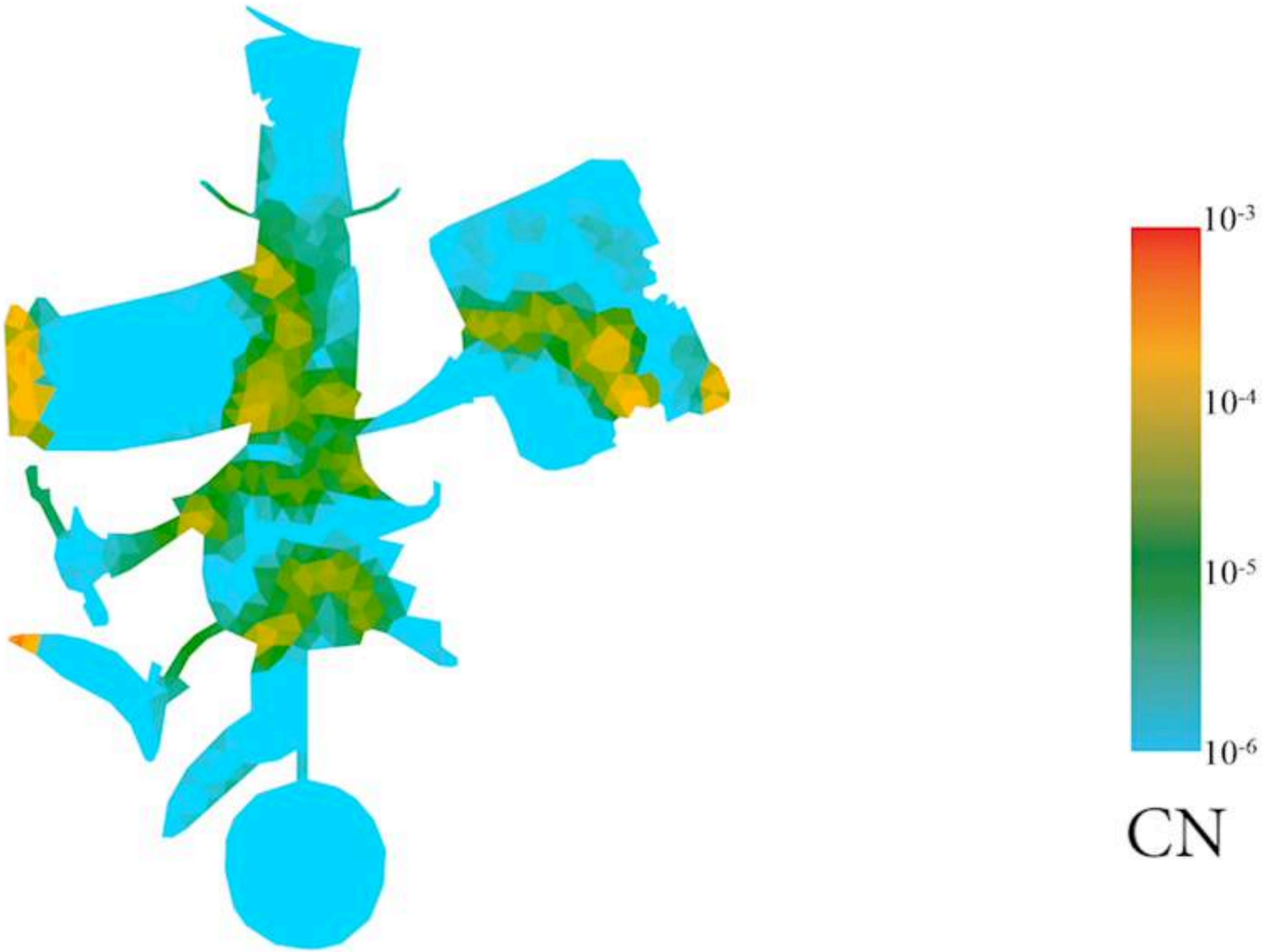
Before DOT iterations:



DOT Iteration Process



DOT iterations:



Elf tests

DOT

PN

L-BFGS-PD

63K nodes, 361K elements,

Time step size: 25ms,

Converged to 10^{-5}CN

Horse test

DOT

PN

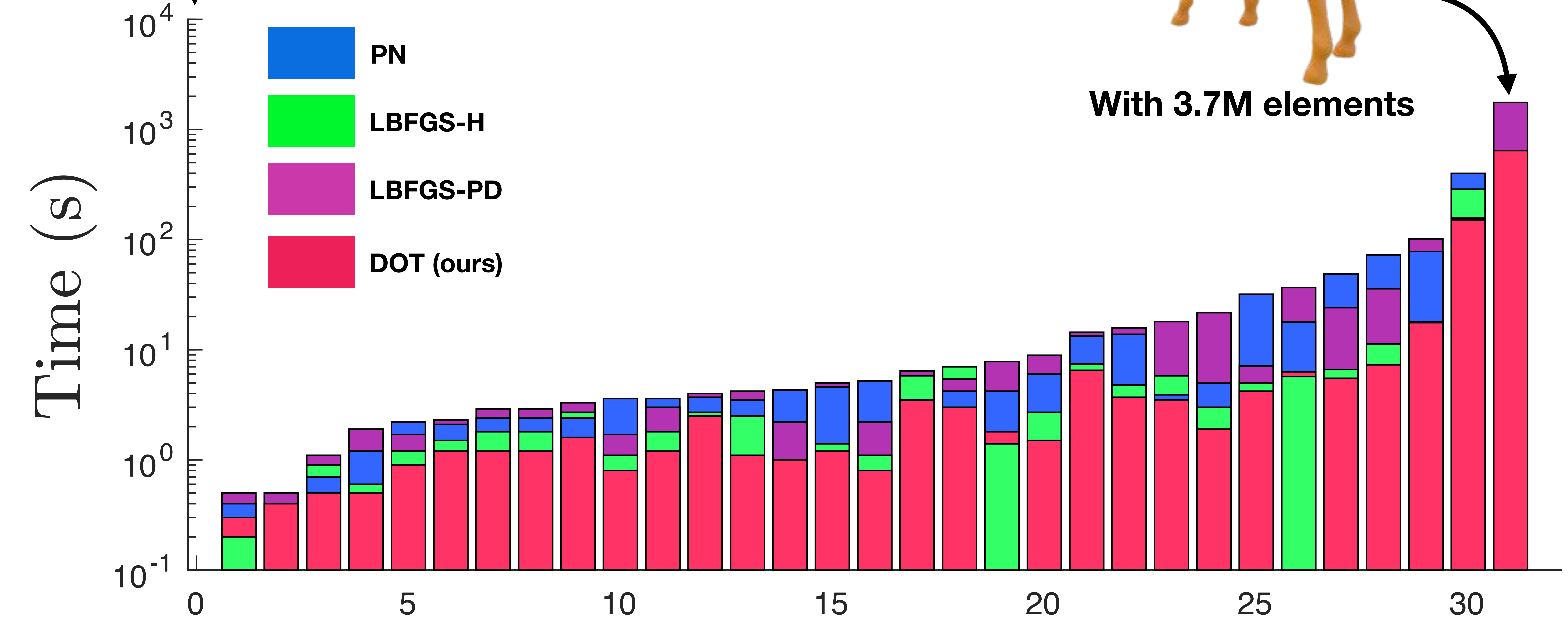
L-BFGS-PD

136K nodes, 642K elements,

Time step size: 25ms,

Converged to 10^{-5}CN

Performance



With 3.7M elements

100K tetrahedra,

Time step size: 10ms,

Converged to 10^{-5}CN

1.9 sec/frame,

147K tetrahedra,

Time step size: 10ms,

Converged to 10^{-5}CN

3.7 sec/frame,

Conclusion

DOT, optimization time step solver that enables

Robust, efficient, and accurate **frame-size time stepping** for challenging **large and high-speed deformations** with **nonlinear materials**.


Thanks!
(Source code coming soon)