Automated Mesh Generation using Graph Neural Networks and Reinforcement Learning

Per-Olof Persson

Department of Mathematics, University of California, Berkeley
Mathematics Department, Lawrence Berkeley National Laboratory

with Arjun Narayanan, Yulong (Lewis) Pan, Will Thacher

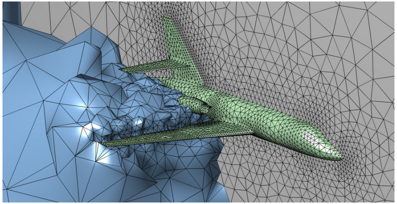
MUSA Math Monday, Department of Mathematics, UC Berkeley

January 26, 2026

From Heuristics to Learning: The Meshing Challenge

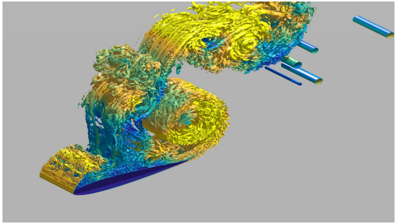
Why Meshing Matters

- Discretizes domains for numerical simulation.
- Crucial for PDEs, Fluid Dynamics, and Graphics.
- Quality dictates **accuracy** and **stability**.



Limitations of Classical Algorithms

- Standard methods (e.g., Delaunay) are rigid.
- Rely on complex, human-designed heuristics.
- Complex geometries often require manual tuning.



The Machine Learning Paradigm

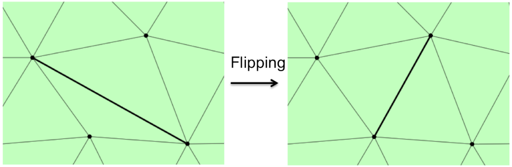
- Can we replace fixed rules with learned policies?
- Treat generation as a **Sequential Decision Process**.
- Goal: Train an RL agent to “play the game.”

Two Strategies for Learning Meshes

We explore two different “games” for the Reinforcement Learning agent:

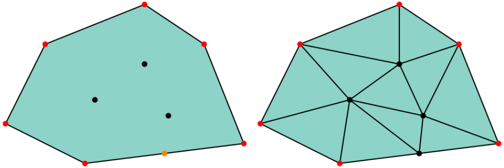
Part I: Topology Optimization (The Connectivity Game)

- **Focus:** Optimizing the graph structure.
- **Actions:** Edge flips and topological moves.
- **Geometry:** Vertex positions are secondary.
- **Goal:** Perfect node regularity (valency).
- **Result:** Structured Quad and Tri meshes.



Part II: Node Placement Strategy (The Geometry Game)

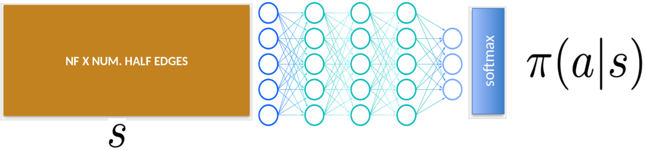
- **Focus:** Optimizing vertex distribution.
- **Actions:** Continuous move, insert, delete.
- **Topology:** Handled by Delaunay algorithm.
- **Goal:** Optimal resolution and sizing.
- **Result:** Adaptive meshes for 2D domains.

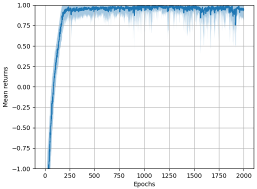


Part I: Topology Optimization

Deep Reinforcement Learning for Block Meshing

- Define a “game” for automatic block mesh improvement:
  - “Moves”: Local or global topological operations (e.g. “flips”)
  - “Score”: Measure of irregularity of the mesh $s = \sum_i |\Delta_i|$
- Use a half-edge mesh structure to define a CNN-type network which extends to fully unstructured quadrilateral meshes
- Train on random geometries, using the PPO algorithm on GPUs
- Consistently produces close-to-optimal meshes

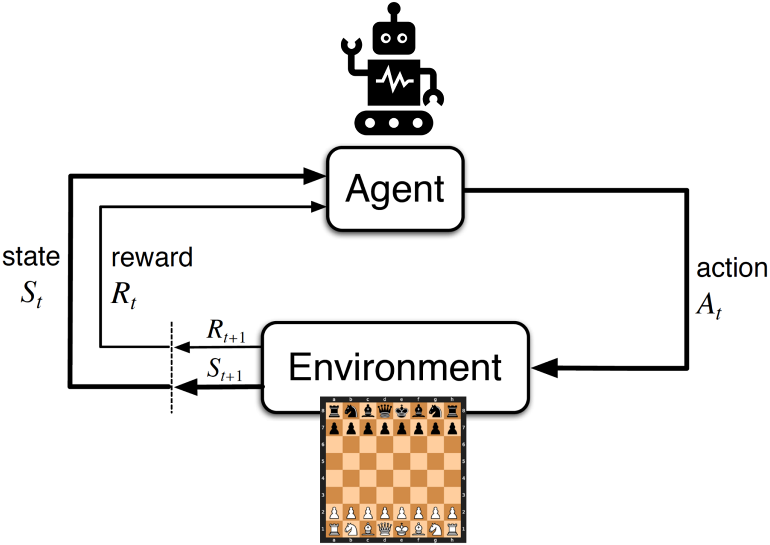




[1] Narayanan, Pan, Persson. *Learning topological operations on meshes with application to block decomposition of polygons*. Computer-Aided Design, Vol. 175, pp. 103744 (2024). arXiv:2309.06484.

Live Mesh Demo

Basic idea of reinforcement learning



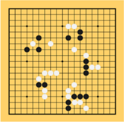
Reinforcement Learning, Solutions Methods

Finite state-space ✓

Finite action-space ✓

Tabular methods
Iterative methods with provable convergence

10^{45} states



10^{170} states

Sampling based methods

- Monte Carlo Tree Search
- Deep RL

Some Useful Terminology

$$\Pi(a_t|s_t, \boxed{\theta})$$

Policy: Probability distribution over actions

$$P(s_{t+1}|s_t, a_t)$$

State transition probability

$$\tau = s_0, a_0, \ldots, s_H, a_H$$

State – action trajectory

$$\boxed{R(\tau) = \sum_{t=0}^{H} R(s_t, a_t)}$$

Cumulative returns of trajectory

Objective function

$$
\begin{aligned}
U(\theta) &= \mathbb{E}\left[R(\tau); \Pi_\theta\right] \\
&= \sum_\tau P(\tau;\theta)R(\tau)
\end{aligned}
$$

$$\theta^* = \arg\max_\theta U(\theta)$$

Estimating gradient of objective

$$U(\theta) = \sum_{\tau} P(\tau;\theta) R(\tau)$$

$$\nabla_\theta U(\theta) = \sum_{\tau} \nabla_\theta P(\tau;\theta) R(\tau)$$

$$= \sum_{\tau} P(\tau;\theta) \frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)} R(\tau)$$

$$= \mathbb{E}\left[\nabla_\theta \log(P(\tau;\theta)) R(\tau)\right] \approx \frac{1}{m} \sum_{i=1}^{m} \boxed{\nabla_\theta \log(P(\tau^{(i)};\theta))} R(\tau^{(i)})$$

Mesh editing operations - triangles

Edge-flip

Edge-split

Collapse

Mesh editing operations - quadrilaterals, local

Flip

Split-Collapse

Mesh editing operations - quadrilaterals, global

Global Split

Global Cleanup

Objective: minimize vertex irregularity

Given:

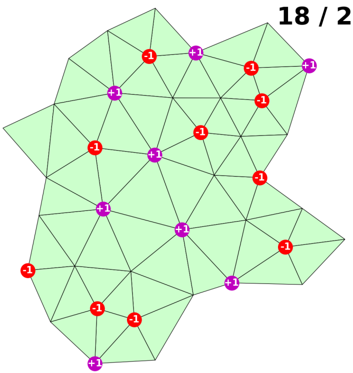
- Mesh m
- Desired degree of vertices d^*:

$$d^* = \begin{cases} 360/\alpha & \text{interior vertex} \\ \max\left(\lfloor \theta/\alpha \rceil + 1, 2\right) & \text{boundary vertex} \end{cases}$$

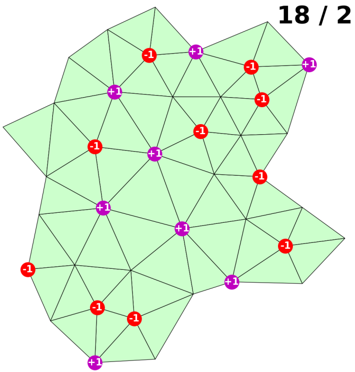
  where $\alpha = 60$ for triangles, 90 for quads, and θ is the angle of a boundary point.

- Define $\Delta_i = d_i - d_i^*$

minimize $s = \sum_i |\Delta_i|$



Lower bound on objective function



Note that:

- $s^* = \left|\sum_i \Delta_i\right| \leq \sum_i |\Delta_i| = s$
- s^* is invariant under mesh edits.

This means s^* is a bound on the best possible improved mesh $\Longrightarrow$ use for a normalized optimality score.

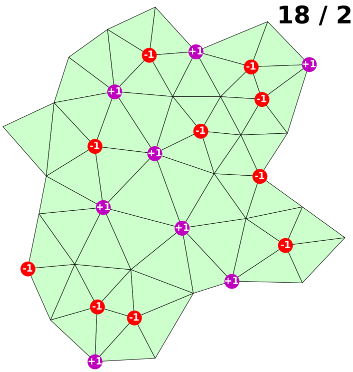
Challenging, unstructured problem

The problem poses several challenges:

- Discrete decisions
- Fully unstructured
- Dynamic data-structure

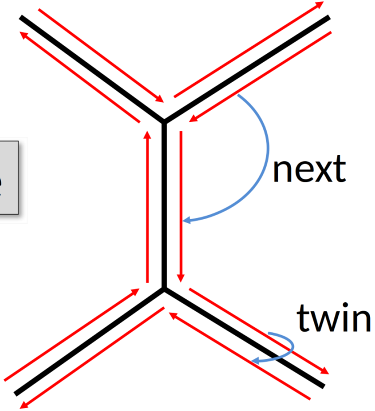
Solution methods need to be able to:

- Represent and understand mesh topology
- Efficiently implement mesh edits



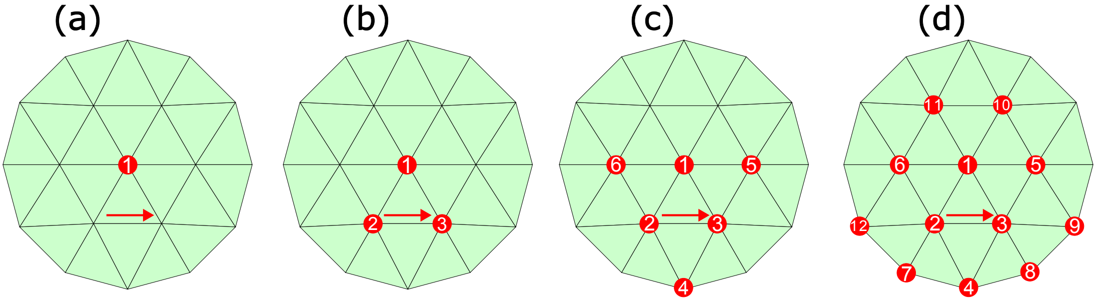
Half-edges represent topology in a structured way

Action: Half-edge + type



Half-edge operations used to represent state

Template: Ordered sequence of vertices around each half-edge



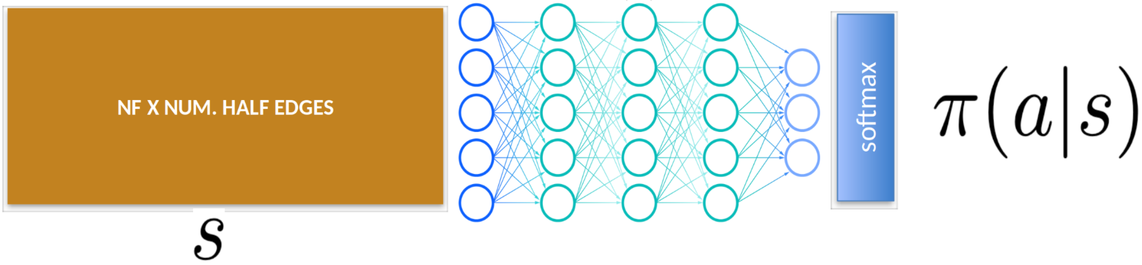
In the language of reinforcement learning

- **State**: Irregularity and degree of vertices in template
- **Action**: Flip, split, collapse, etc.
- **Reward**: $r_t = s_t - s_{t+1}$

Training procedure:

- Generate random 10-30 sided polygons
- Initial mesh by Delaunay refinement, split using Catmull-Clark for quads
- Terminate if $s^* = s$ or a maximum number of steps taken
- Monitor normalized returns

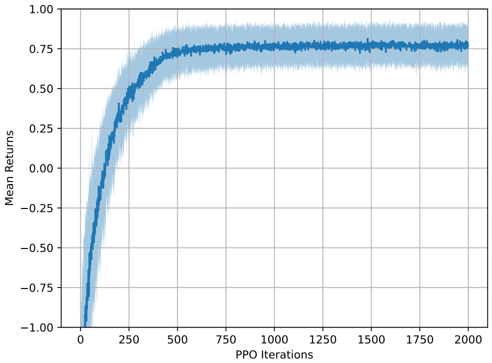
Neural network learns a mesh edit policy



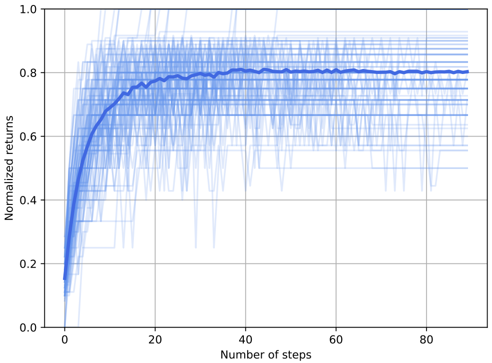
Trained in self-play by Proximal Policy Optimization (PPO) algorithm

Schulman, John, et al. *Proximal policy optimization algorithms* arXiv:1707.06347 (2017).

Results: Triangular Meshes



Average performance over training history



Evaluating the trained agent on multiple rollouts

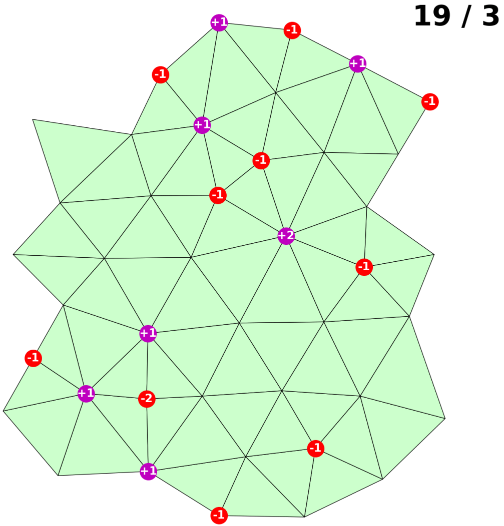
Performance of the triangle mesh agent over the training history.

Results: Triangular Meshing

Triangular meshing

Example 1

Step 0 (out of 27)

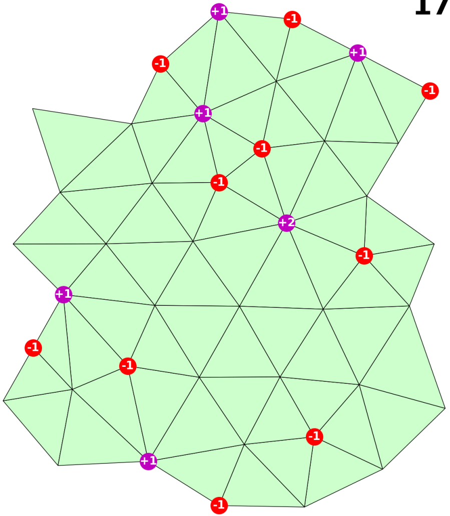


Results: Triangular Meshing

Triangular meshing

Example 1

Step 1 (out of 27)

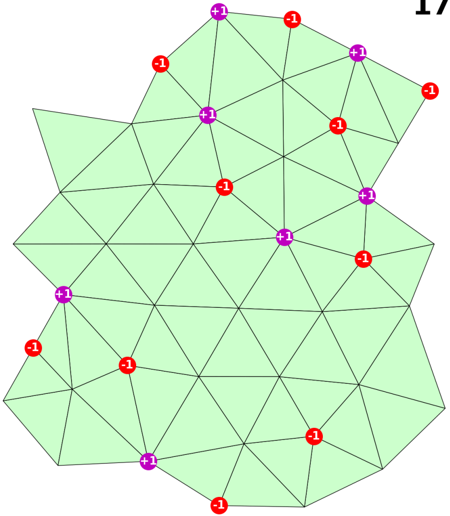


Results: Triangular Meshing

Triangular meshing

Example 1

Step 2 (out of 27)

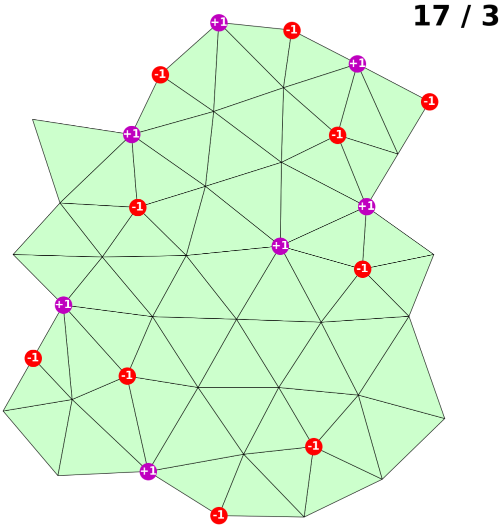


Results: Triangular Meshing

Triangular meshing

Example 1

Step 3 (out of 27)

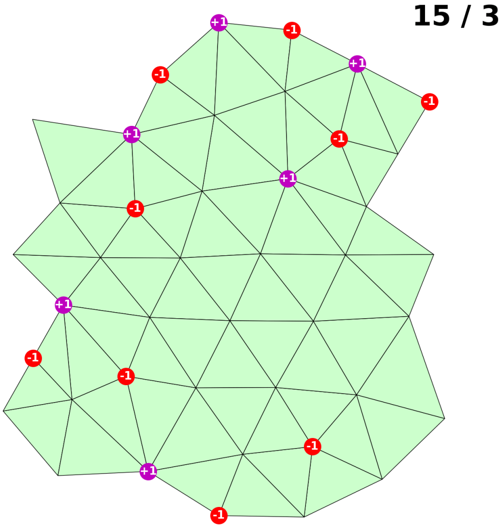


Results: Triangular Meshing

Triangular meshing

Example 1

Step 4 (out of 27)

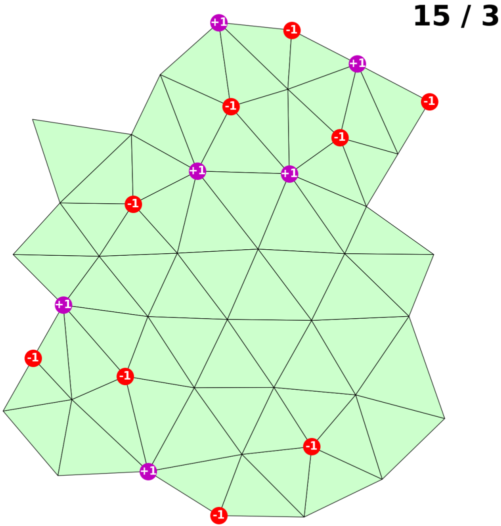


Results: Triangular Meshing

Triangular meshing

Example 1

Step 5 (out of 27)

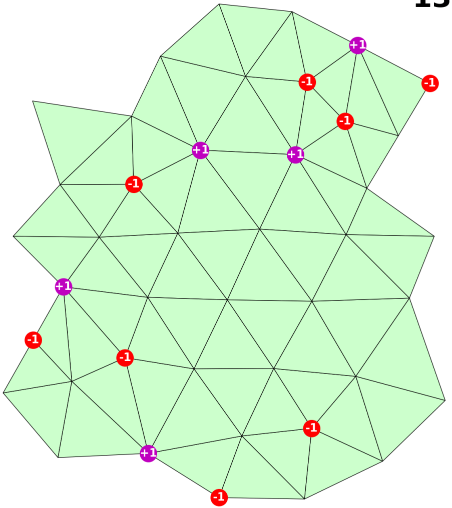


Results: Triangular Meshing

Triangular meshing

Example 1

Step 6 (out of 27)

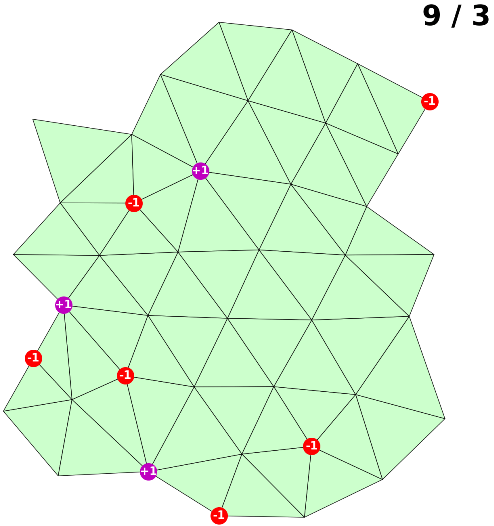


Results: Triangular Meshing

Triangular meshing

Example 1

Step 7 (out of 27)

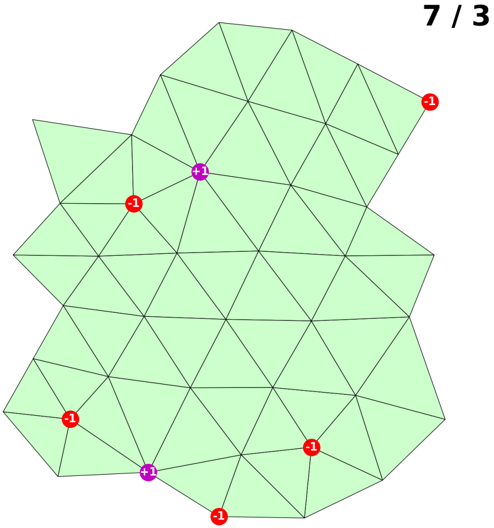


Results: Triangular Meshing

Triangular meshing

Example 1

Step 8 (out of 27)

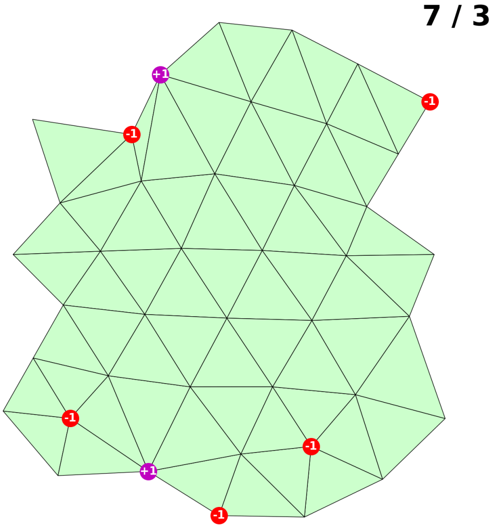


Results: Triangular Meshing

Triangular meshing

Example 1

Step 9 (out of 27)

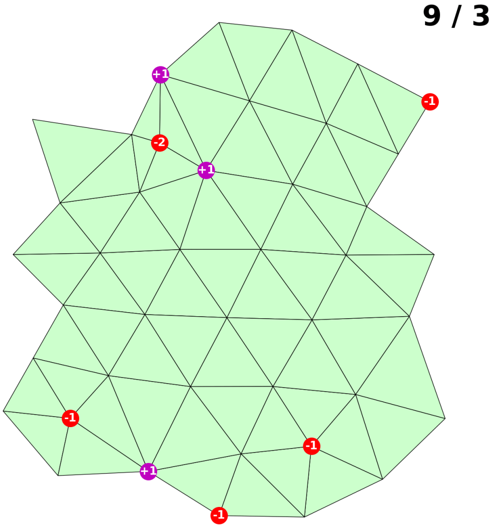


Results: Triangular Meshing

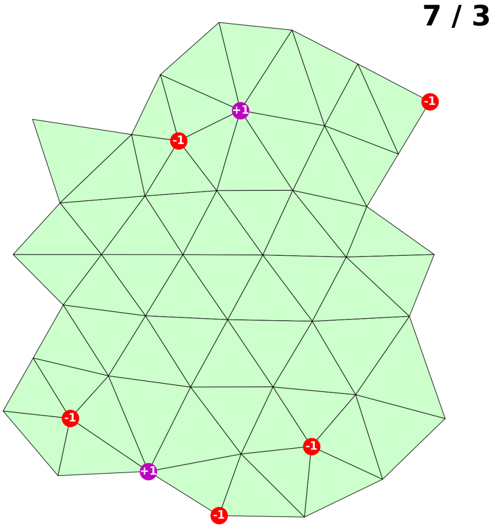
Triangular meshing

Example 1

Step 10 (out of 27)



Results: Triangular Meshing



Triangular meshing

Example 1

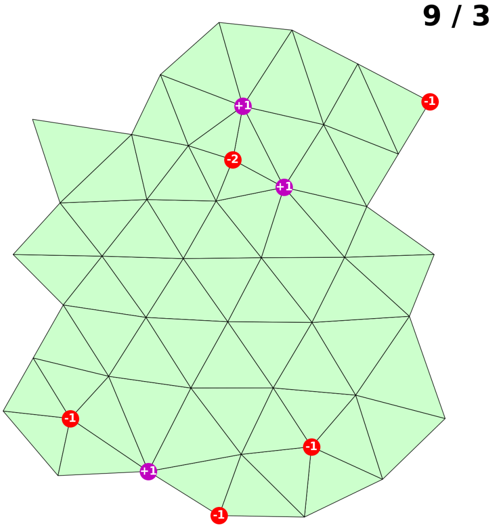
Step 11 (out of 27)

Results: Triangular Meshing

Triangular meshing

Example 1

Step 12 (out of 27)

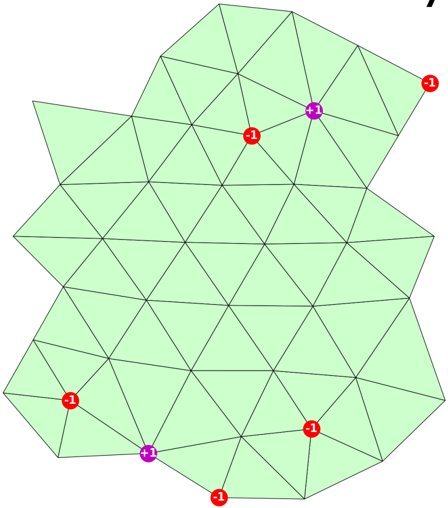


Results: Triangular Meshing

Triangular meshing

Example 1

Step 13 (out of 27)

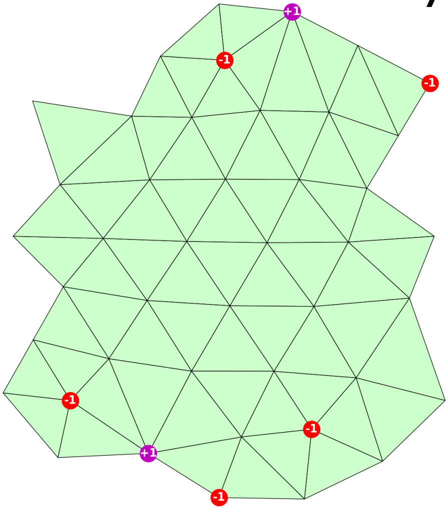


Results: Triangular Meshing

Triangular meshing

Example 1

Step 14 (out of 27)

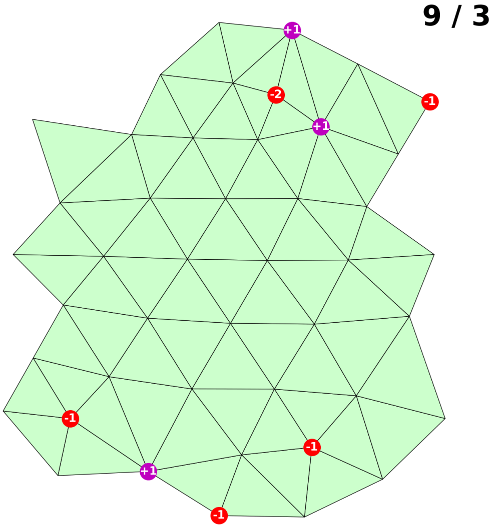


Results: Triangular Meshing

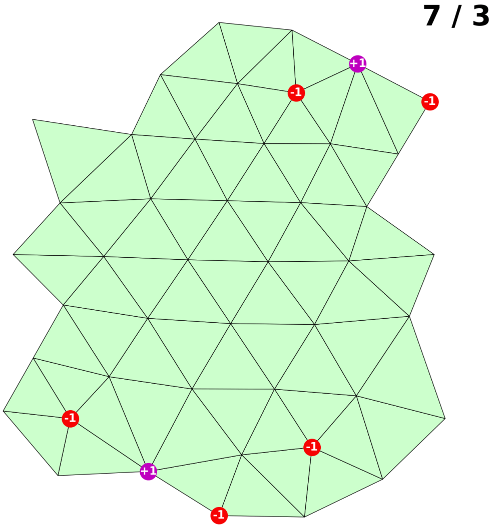
Triangular meshing

Example 1

Step 15 (out of 27)



Results: Triangular Meshing



Triangular meshing

Example 1

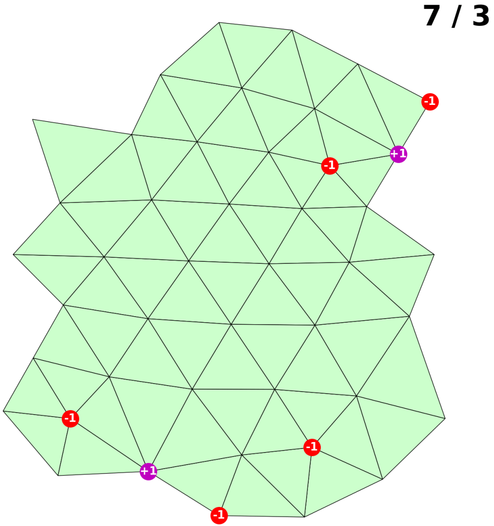
Step 16 (out of 27)

Results: Triangular Meshing

Triangular meshing

Example 1

Step 17 (out of 27)

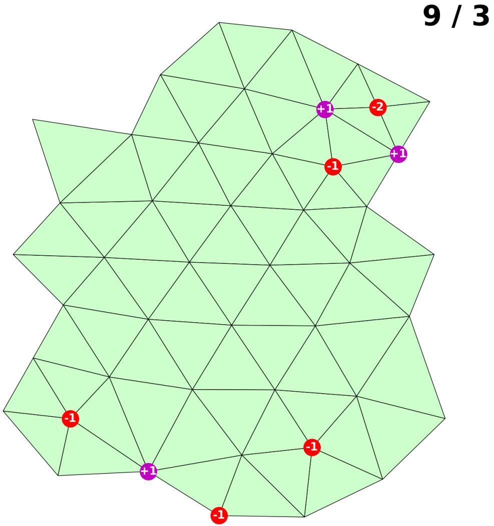


Results: Triangular Meshing

Triangular meshing

Example 1

Step 18 (out of 27)

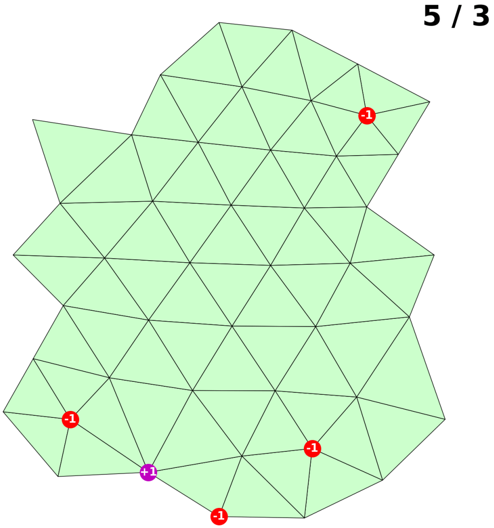


Results: Triangular Meshing

Triangular meshing

Example 1

Step 19 (out of 27)

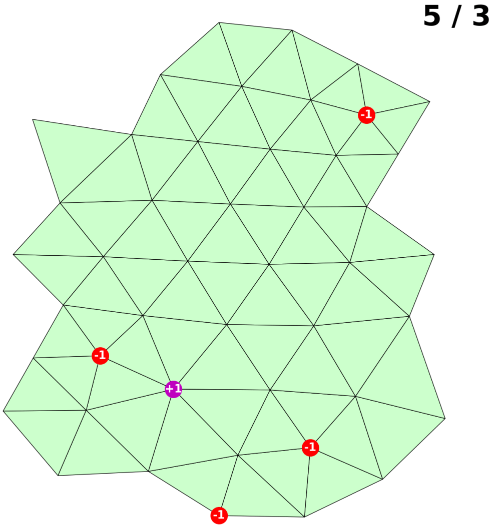


Results: Triangular Meshing

Triangular meshing

Example 1

Step 20 (out of 27)

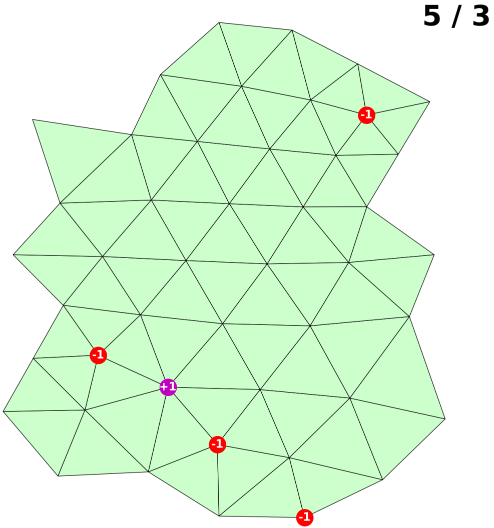


Results: Triangular Meshing

Triangular meshing

Example 1

Step 21 (out of 27)

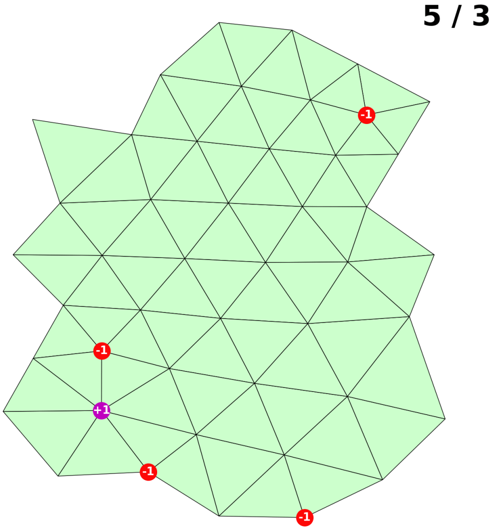


Results: Triangular Meshing

Triangular meshing

Example 1

Step 22 (out of 27)

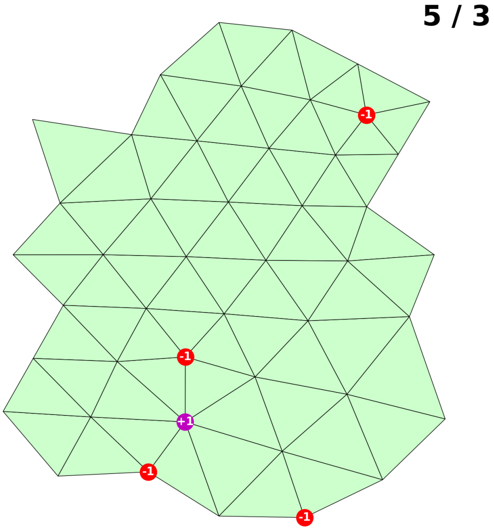


Results: Triangular Meshing

Triangular meshing

Example 1

Step 23 (out of 27)



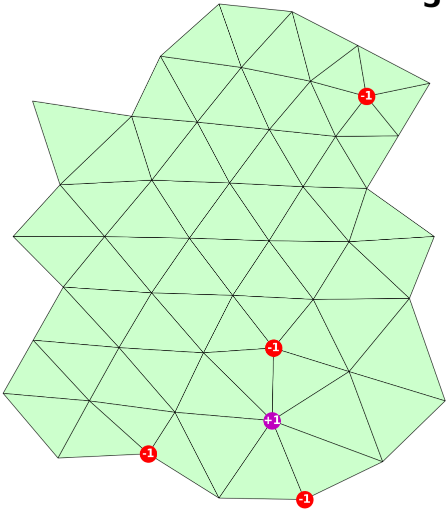
Results: Triangular Meshing

5 / 3

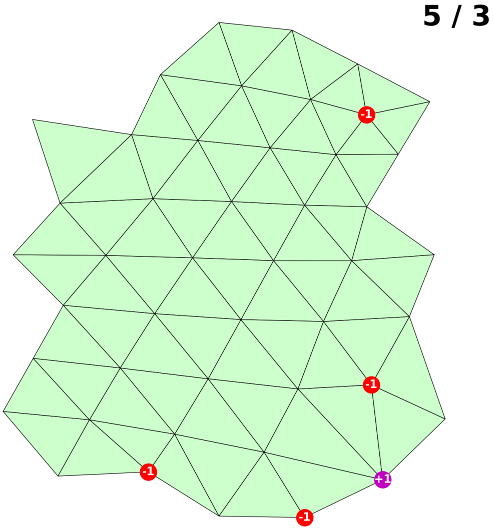
Triangular meshing

Example 1

Step 24 (out of 27)



Results: Triangular Meshing



Triangular meshing

Example 1

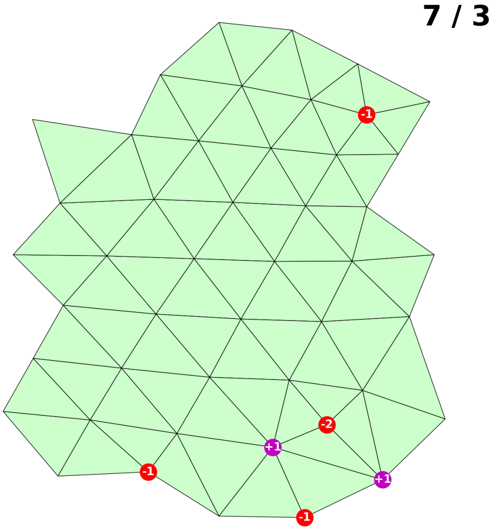
Step 25 (out of 27)

Results: Triangular Meshing

Triangular meshing

Example 1

Step 26 (out of 27)

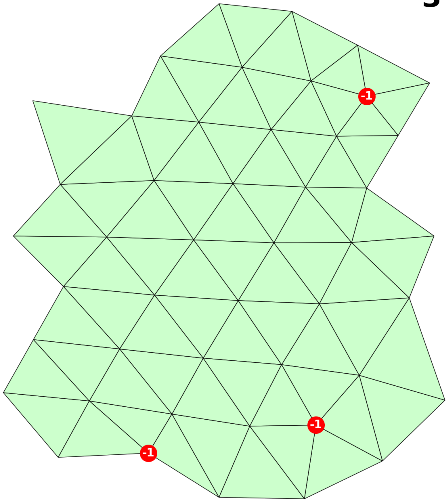


Results: Triangular Meshing

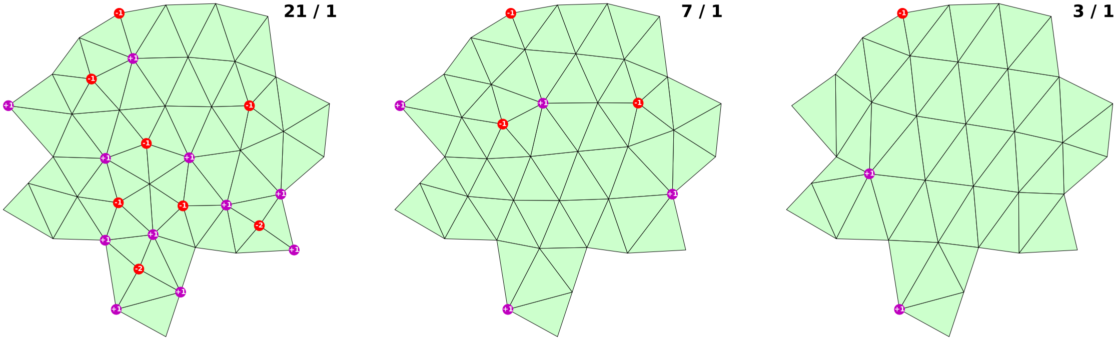
Triangular meshing

Example 1

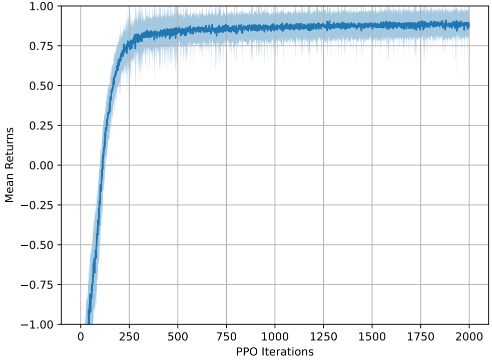
Step 27 (out of 27)



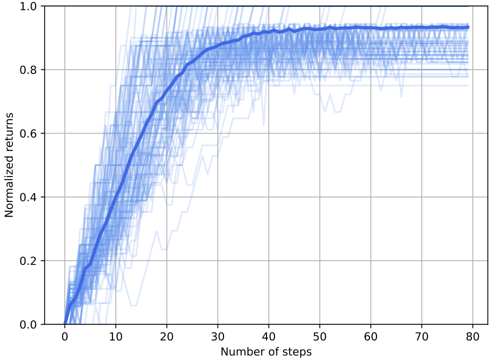
Triangular meshing example: 20-sided polygon



Results: Quadrilateral Meshes



Average performance over training history



Evaluating the trained agent on multiple rollouts

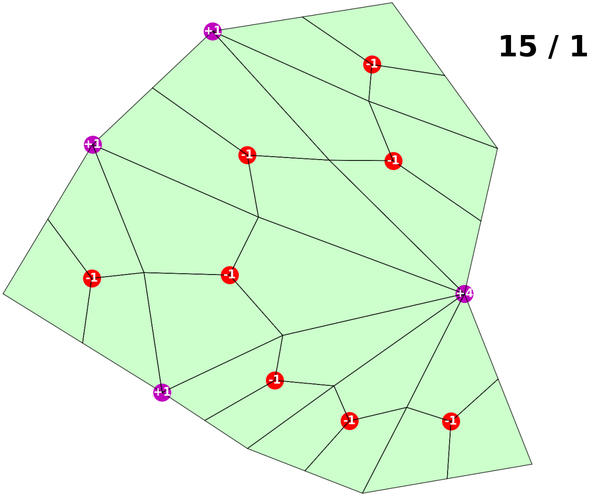
Performance of the quadrilateral mesh agent over the training history.

Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 0 (out of 19)

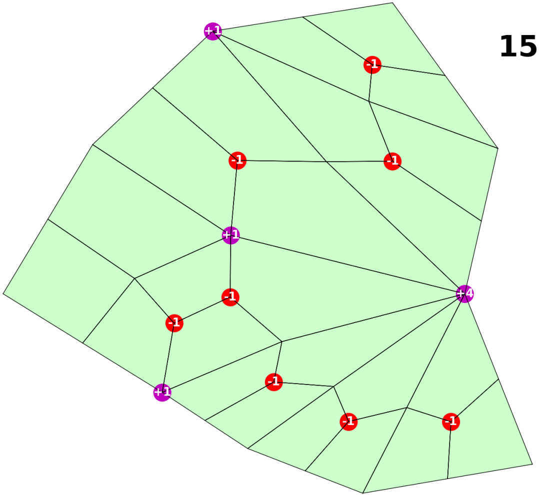


Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 1 (out of 19)



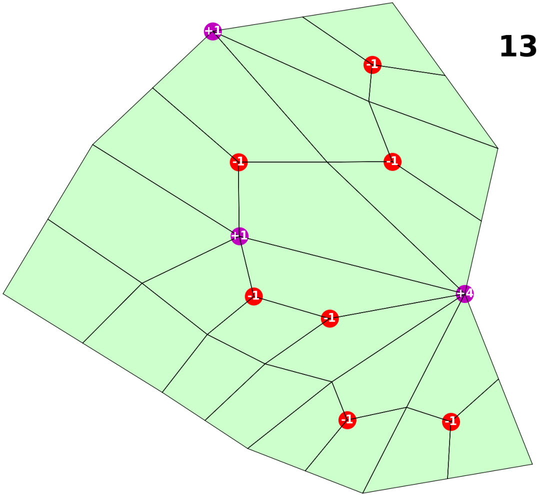
15 / 1

Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 2 (out of 19)



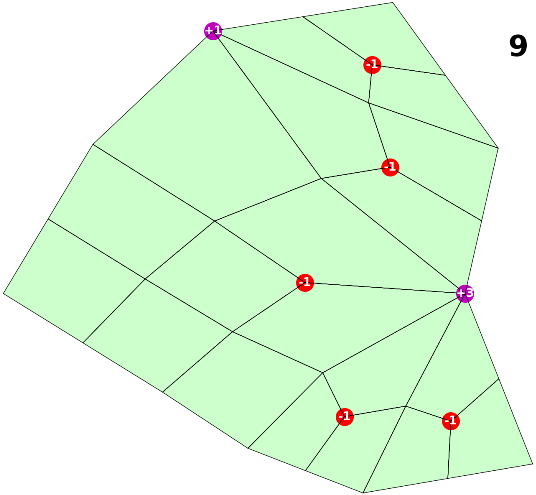
13 / 1

Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 3 (out of 19)



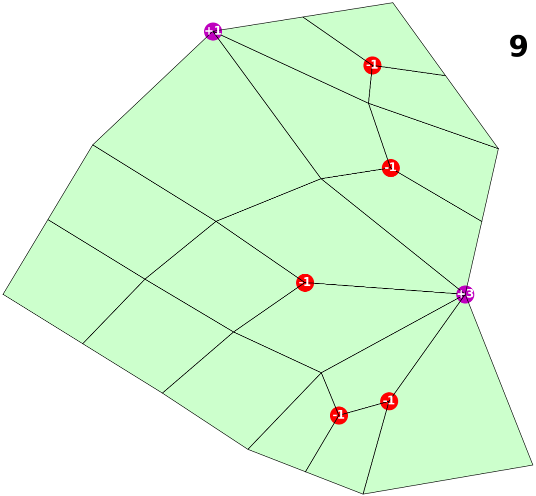
9 / 1

Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 4 (out of 19)



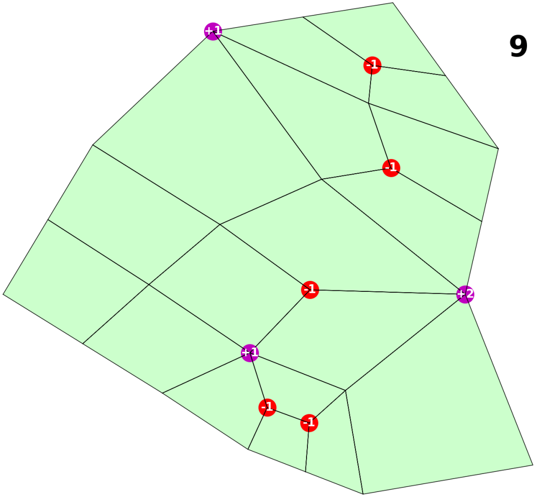
9 / 1

Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 5 (out of 19)



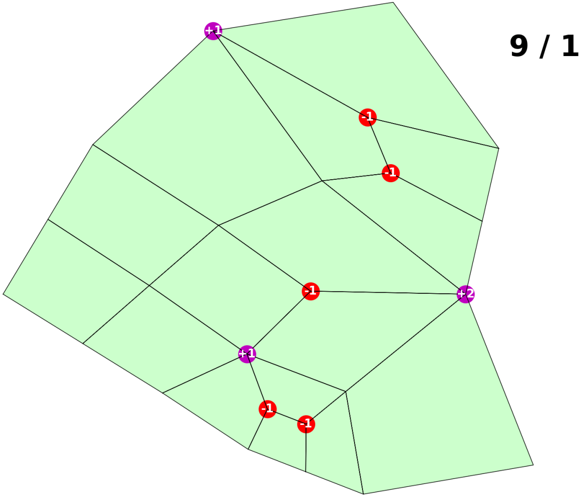
9 / 1

Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 6 (out of 19)

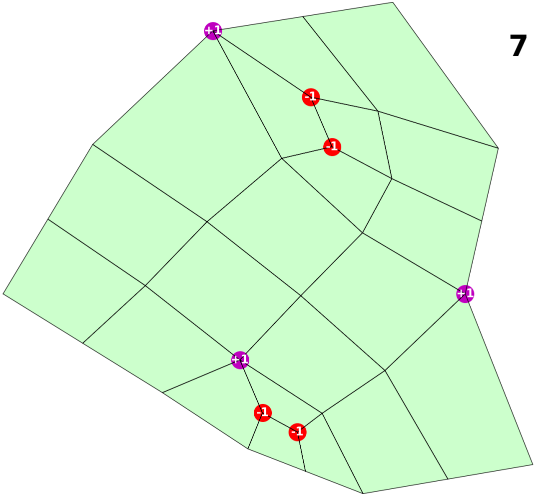


Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 7 (out of 19)



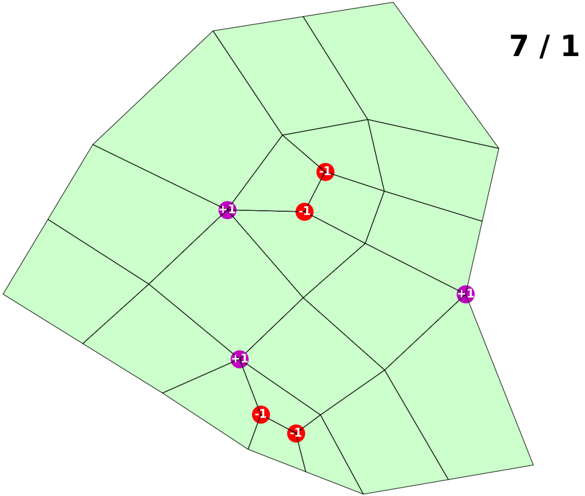
7 / 1

Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 8 (out of 19)

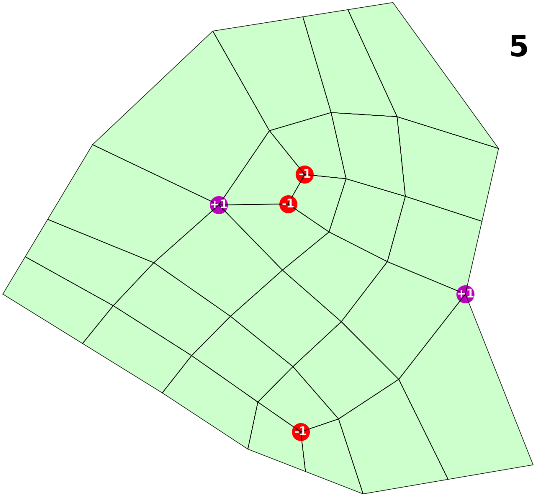


Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 9 (out of 19)



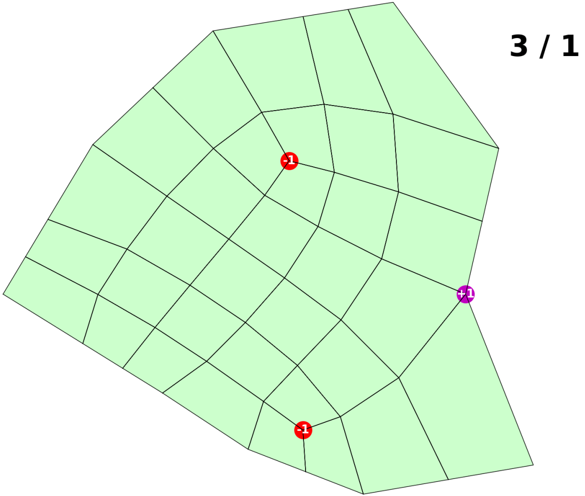
5 / 1

Results: Quadrilateral block meshing

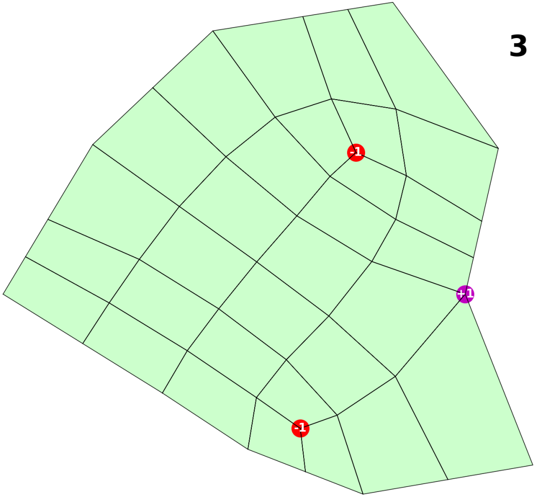
Block mesh decomposition

Example 1

Step 10 (out of 19)



Results: Quadrilateral block meshing



3 / 1

Block mesh decomposition

Example 1

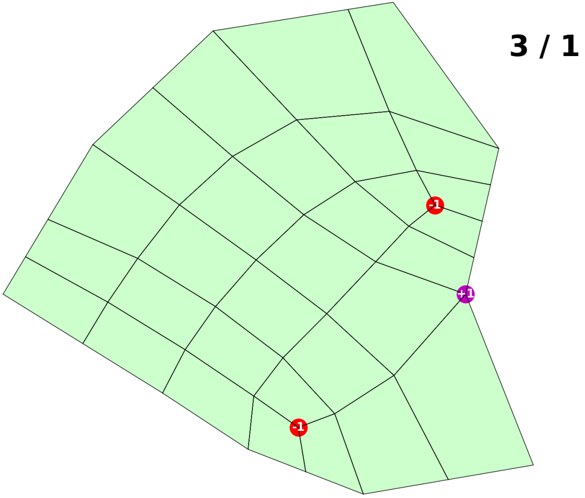
Step 11 (out of 19)

Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 12 (out of 19)

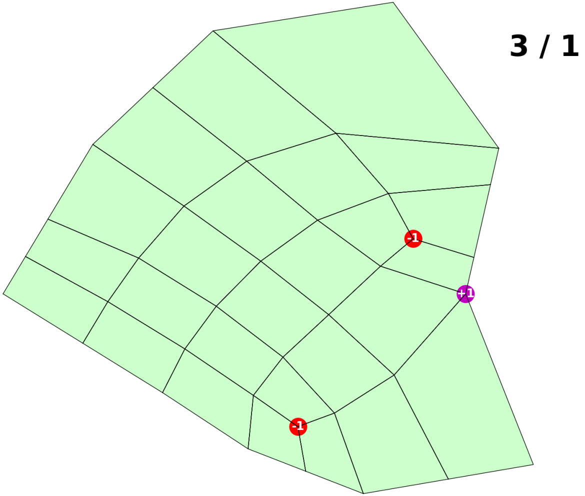


Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 13 (out of 19)

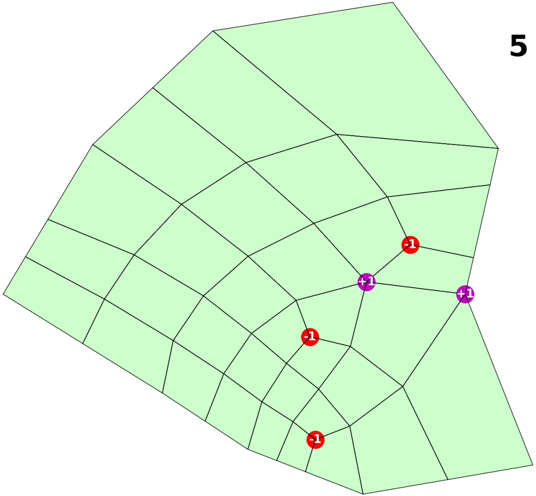


Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 14 (out of 19)



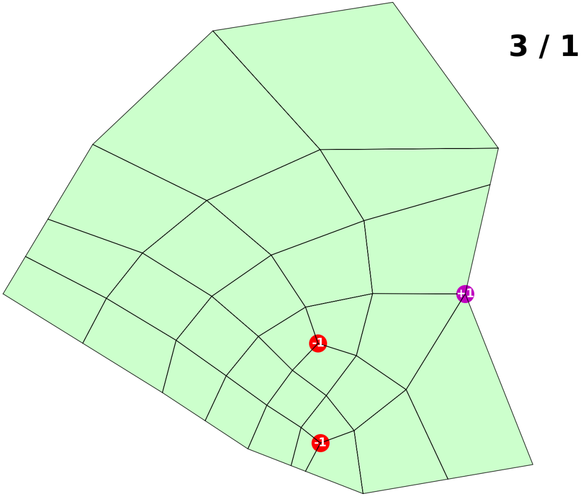
5 / 1

Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 15 (out of 19)

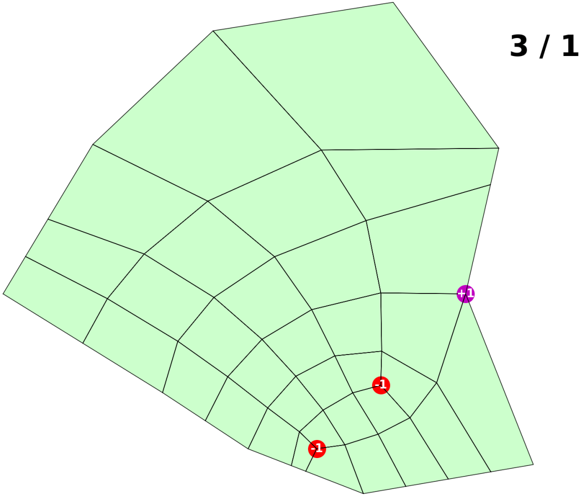


Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 16 (out of 19)

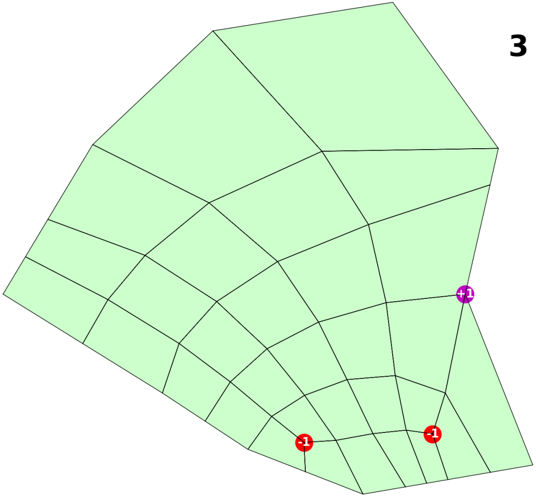


Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 17 (out of 19)



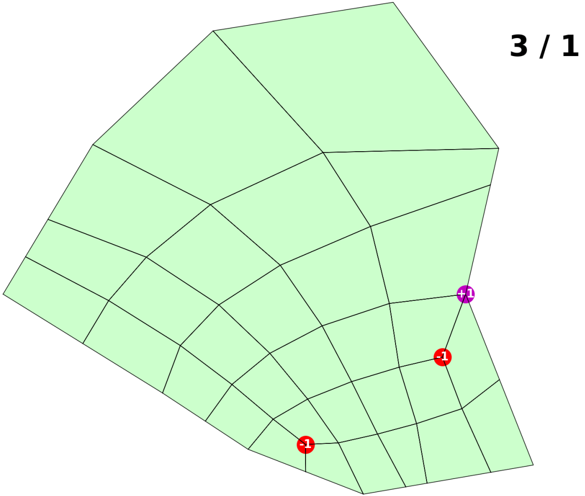
3 / 1

Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 18 (out of 19)

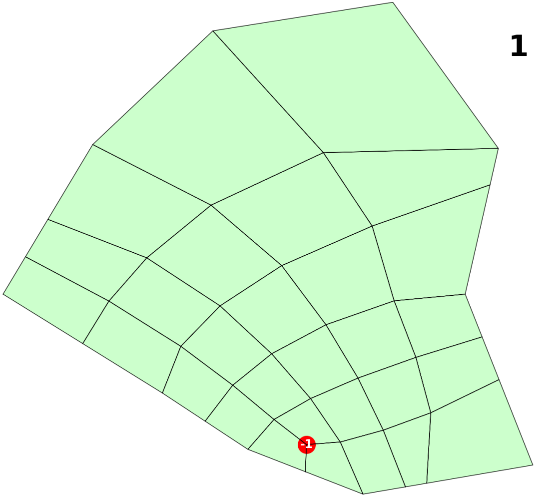


Results: Quadrilateral block meshing

Block mesh decomposition

Example 1

Step 19 (out of 19)



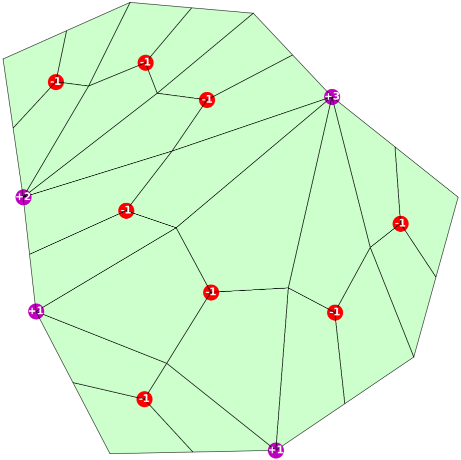
1 / 1

Results: Quadrilateral block meshing

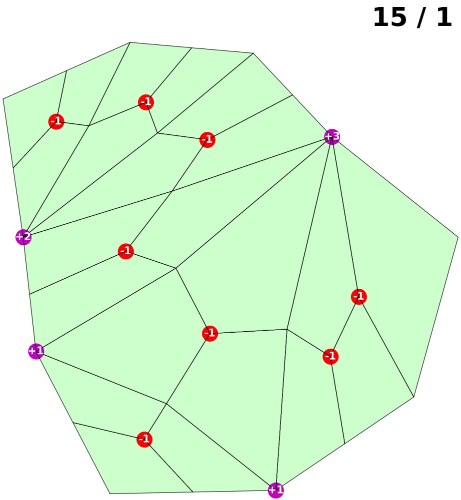
Block mesh decomposition

Example 2

Step 0 (out of 12)



Results: Quadrilateral block meshing



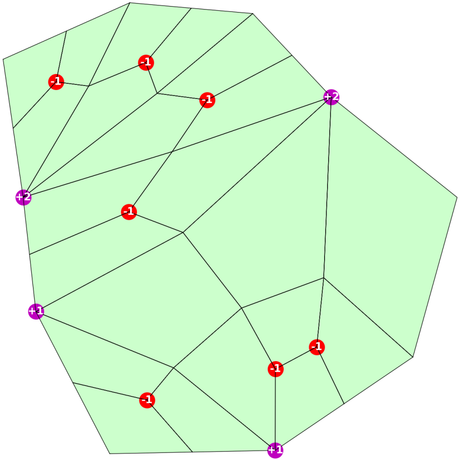
Block mesh decomposition

Example 2

Step 1 (out of 12)

Results: Quadrilateral block meshing

13 / 1



Block mesh decomposition

Example 2

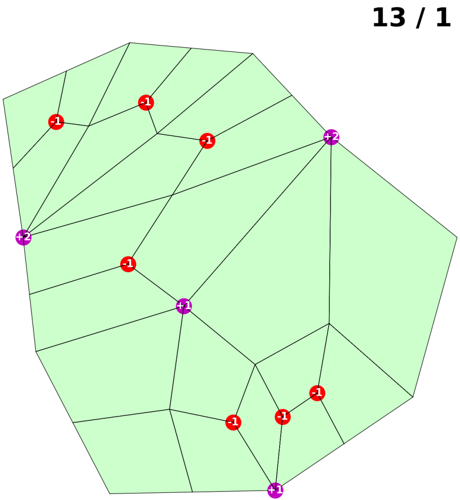
Step 2 (out of 12)

Results: Quadrilateral block meshing

Block mesh decomposition

Example 2

Step 3 (out of 12)

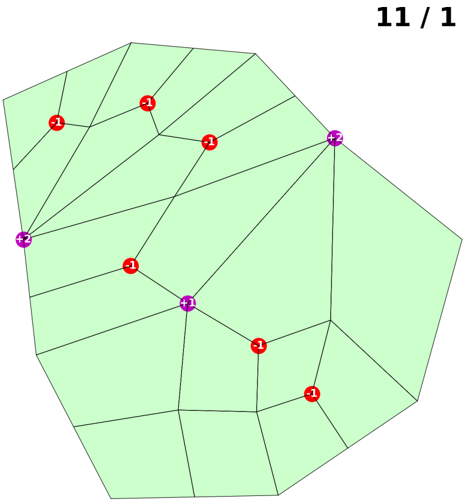


Results: Quadrilateral block meshing

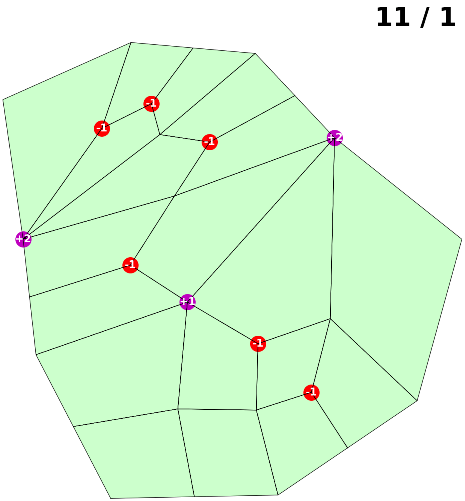
Block mesh decomposition

Example 2

Step 4 (out of 12)



Results: Quadrilateral block meshing



Block mesh decomposition

Example 2

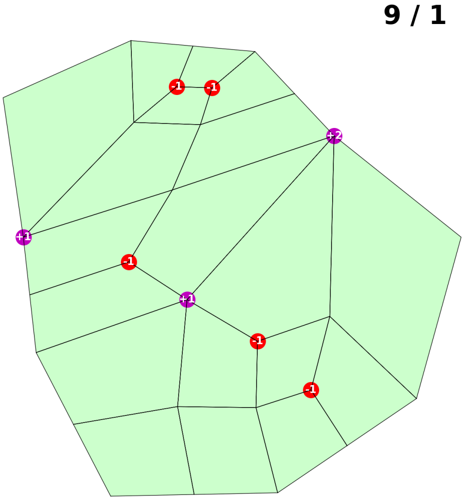
Step 5 (out of 12)

Results: Quadrilateral block meshing

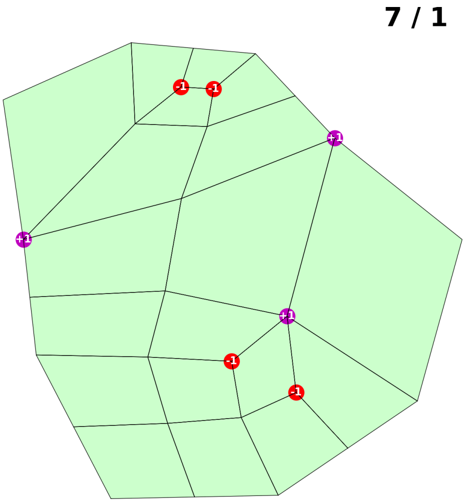
Block mesh decomposition

Example 2

Step 6 (out of 12)



Results: Quadrilateral block meshing



Block mesh decomposition

Example 2

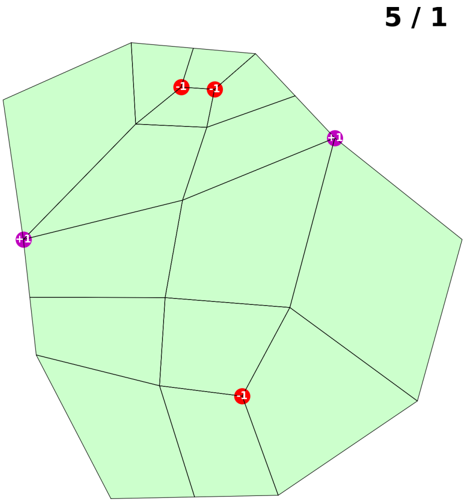
Step 7 (out of 12)

Results: Quadrilateral block meshing

Block mesh decomposition

Example 2

Step 8 (out of 12)

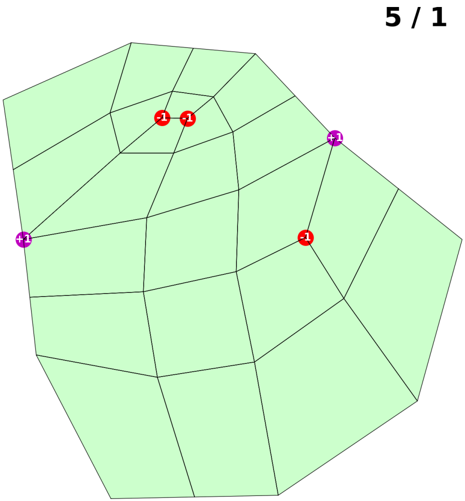


Results: Quadrilateral block meshing

Block mesh decomposition

Example 2

Step 9 (out of 12)

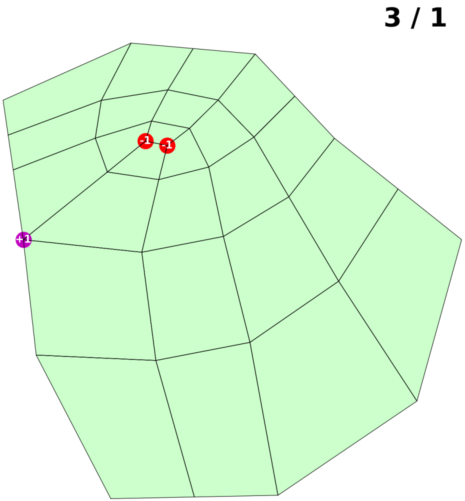


Results: Quadrilateral block meshing

Block mesh decomposition

Example 2

Step 10 (out of 12)

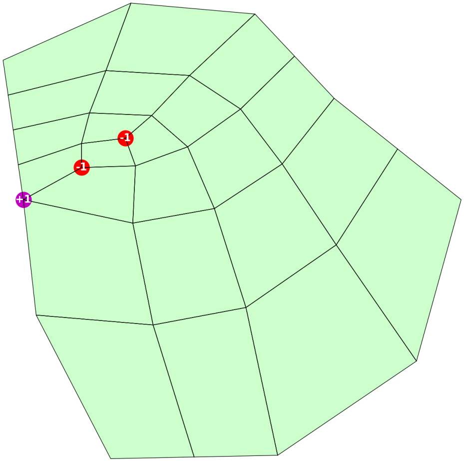


Results: Quadrilateral block meshing

Block mesh decomposition

Example 2

Step 11 (out of 12)

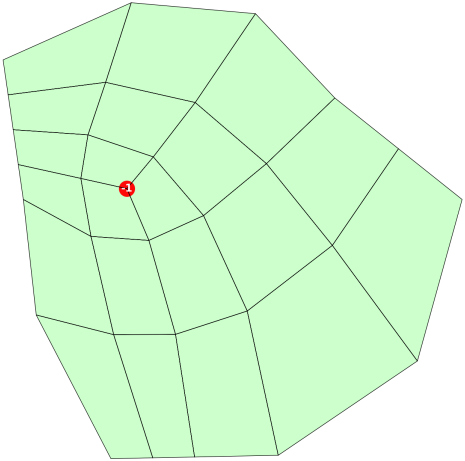


Results: Quadrilateral block meshing

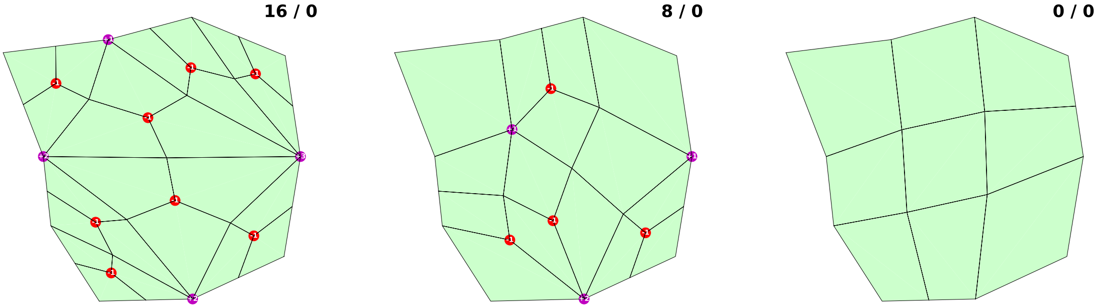
Block mesh decomposition

Example 2

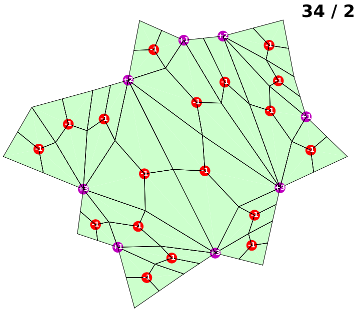
Step 12 (out of 12)

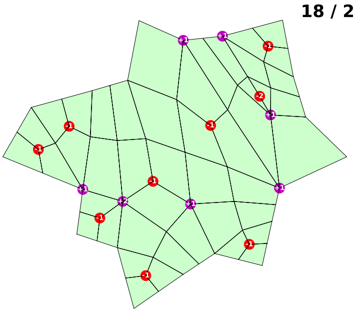


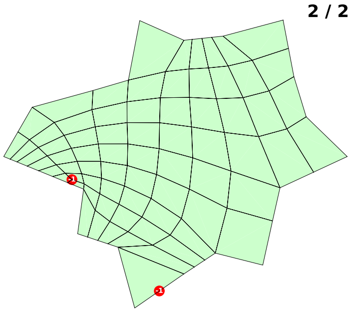
Block decomposition example: 10-sided polygon



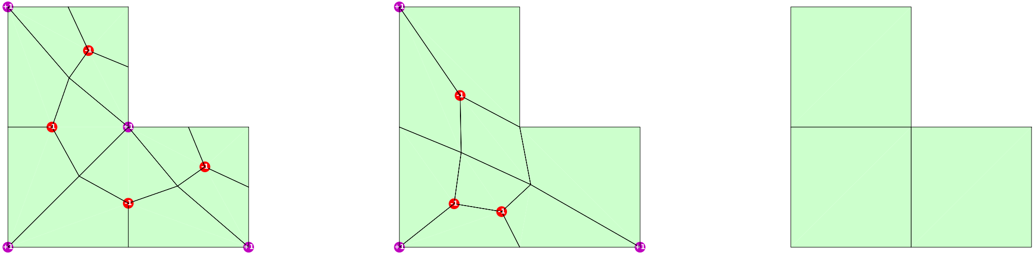
Block decomposition example: 20-sided polygon



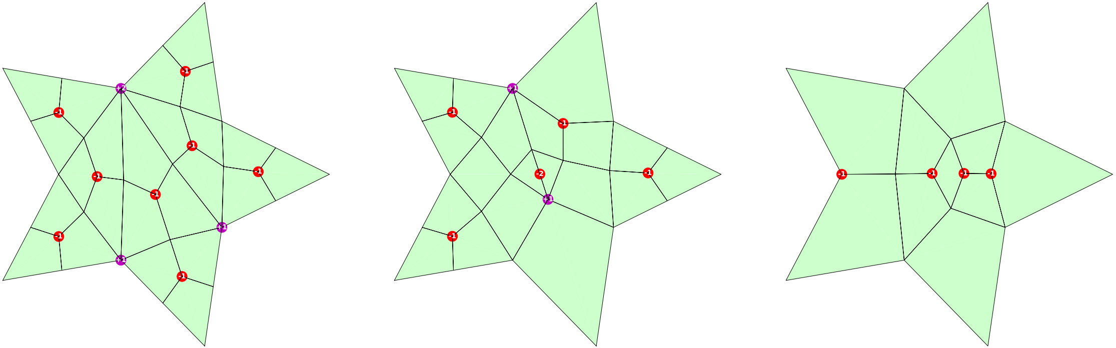




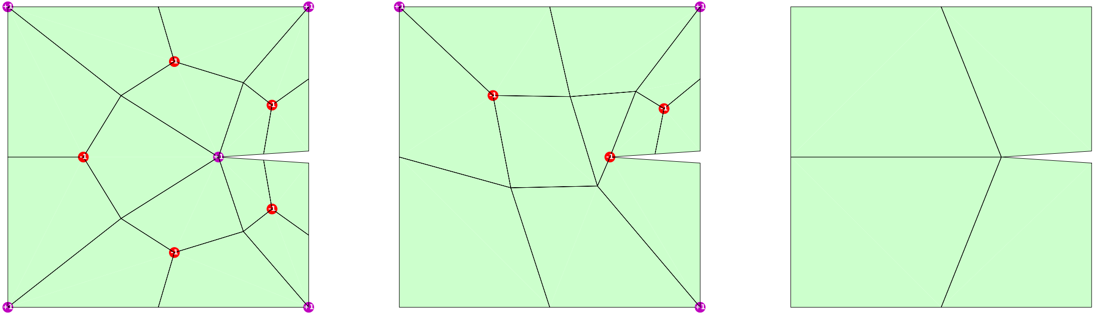
Block decomposition example: L-shaped domain



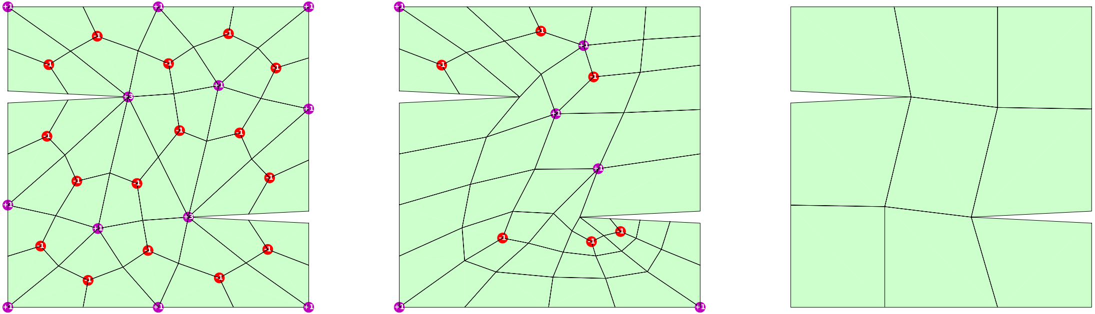
Block decomposition example: Star-shaped domain



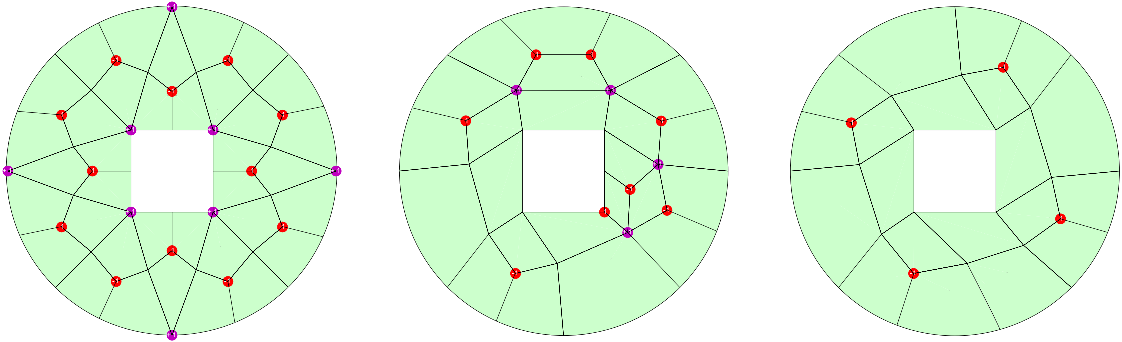
Block decomposition example: Notch domain



Block decomposition example: Double notch domain



Block decomposition example: Square hole in circle domain



Part II: Node Placement Strategy

Extension to ML-based node positions

- Previous work focused on *topology* of the mesh, with node positions determined using some smoothing procedure
- Here, we study the capability of deep networks to also determine the node positions
- Ultimately, the two components should be combined into a complete mesh generator
- For now, we hard-wire the topologies to be Delaunay triangulations

A Learning-Based Approach

- Formulate mesh generation as a sequential decision-making problem.
- Define a parametric strategy (a policy) for mesh operations:
  - Move, add, and delete vertices.
- Use a Graph Neural Network (GNN) with encoder/decoder and convolutional layers
  - Encodes vertex neighborhoods and mesh topology.
  - Outputs vertex modification actions.
- Use reinforcement learning to optimize this strategy.
- Objective: maximize quality metric over generated meshes.

Reinforcement Learning Formulation

- State $s^t = (\mathcal{V}^t, \mathcal{E}^t)$: the mesh at timestep t
- Policy π_θ: maps s^t to a distribution over actions a^t
- Environment applies a^t to get new vertex list $\{x_i^{t+1}\}$ and triangulation $\{T_k^{t+1}\}$
- Sequence $s^0, s^1, \ldots, s^n$ forms a trajectory τ
- Trajectories τ are sampled from a distribution $p_\theta(\tau)$

Action Space

At each timestep, actions may include:

1. Move any non-boundary node
2. Delete any non-boundary node
3. Add new vertex at midpoint of any edge
4. Add new vertex at centroid of any triangle

- Action space depends on the current state s^t
- Policy is stochastic: outputs a distribution over actions

Trajectory Visualization

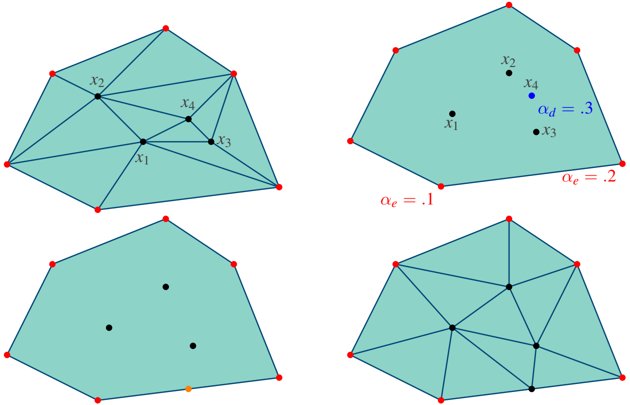


Figure: Mesh generation step: Updated node positions and action variables are sampled from $\pi_\theta(a^t|s^t)$. Nodes are moved, added and deleted accordingly. The node in blue is deleted and the orange node is added at the midpoint of an existing edge.

Mesh Quality Metrics

Each mesh state s^t is evaluated using four quality metrics. Meshes are normalized so that the target edge length is 1 and the target element volume is $\sqrt{3}/4$.

1. Edge length: $q_e(e_{ij}) = 1 - |1 - \|x_i - x_j\||$
2. Angle: $q_a(\gamma_l) = 1 - \frac{|\gamma_l^* - \gamma_l|}{\gamma_l^*}$
3. Volume: $q_v(T_k) = 1 - \frac{|\,|T_k| - \sqrt{3}/4|}{\sqrt{3}/4}$
4. Shape: $q_r(T_k) = 2 \cdot \frac{\rho_{\text{in}}}{\rho_{\text{out}}}$

Reward and Optimization Objective

- Total score $S(s^t)$ = weighted average of metric norms
- Reward at timestep t:

$$r^t = S(s^{t+1}) - S(s^t)$$

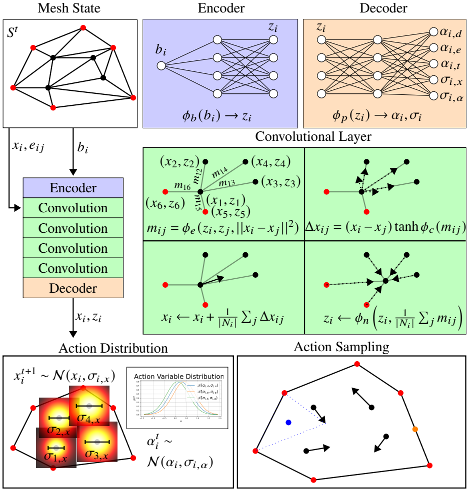
- Trajectory reward:

$$R(\tau) = \sum_{t=1}^{n} r^t = S(s^n)$$

- Objective:

$$\max_{\theta} \mathbb{E}_{\Omega \sim p_D} \left[\mathbb{E}_{\tau \sim p_\theta} R(\tau) \right]$$

Mesh Generator Architecture



Overview of the mesh generator architecture.

- Mesh is represented as a graph:
  - Vertices $\rightarrow$ nodes
  - Edges $\rightarrow$ edges
- GNN modifies the vertex set:
  - Moves existing vertices
  - Adds or deletes vertices
- Delaunay triangulation creates updated mesh
- Process iterated multiple times
- GNN trained via reinforcement learning to optimize mesh quality

Graph Neural Network: Convolution

Node updates at each layer:

$$m_{ij} = \phi_e(z_i, z_j, \|x_i - x_j\|^2)$$

$$\Delta x_{ij} = (x_i - x_j) \tanh \phi_c(m_{ij})$$

$$x_i \leftarrow x_i + \mathbf{1}_0(b_i) \cdot \frac{1}{|N_i|} \sum_{j \in N_i} \Delta x_{ij}$$

$$z_i \leftarrow \phi_n \left(z_i, \frac{1}{|N_i|} \sum_{j \in N_i} m_{ij} \right)$$

Properties of the GNN Architecture:

- Rotation- and translation-equivariant updates
- Interior equilateral regions remain unchanged
- Inspired by the update rule in DistMesh
- Fully differentiable via automatic differentiation

Decoder and Action Sampling

- Decoder maps feature z_i to:

$$[x_i, \alpha_{i,d}, \alpha_{i,e}, \alpha_{i,t}, \sigma_{i,x}, \sigma_{i,\alpha}]$$

- Sampled from independent Gaussians
- Actions based on thresholds:
  - $\alpha_{i,d} < \nu$: delete node
  - $\alpha_{i,e}, \alpha_{i,t}$: insert edge/triangle point if under ν
- $\nu = 1/4$ in all experiments

Training with PPO

- Alternate between:
  - Sampling trajectories
  - Updating policy parameters θ using PPO
- Optimize *advantage* function $A_\theta(s^t, a_0^t) =: Q(s^t, a_0^t) - V(s^t)$, measuring the difference in expected reward over the remainder of the trajectory between action a_0^t and current policy
- Value function $V_\phi(s^t)$ trained as a GNN with same architecture as policy

Implementation details:

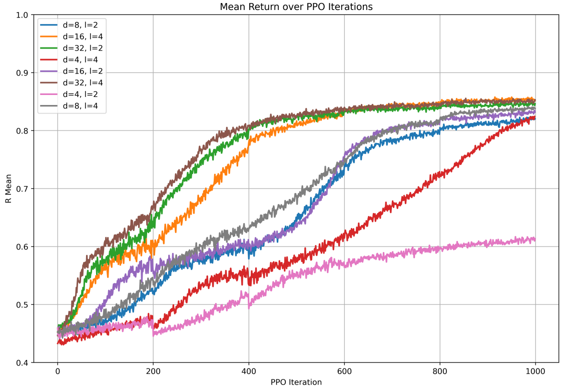
- PPO implementation via RLlib
- GNN built with PyTorch Geometric
- All networks use:
  - One hidden layer MLPs
  - Swish activation
- Single-precision used throughout

Training Curriculum

Training Phase	I	II	III	IV	V
Polygon scaling	2	2.5	3	3.5	4
Num. side range	5-10	5-12	5-14	5-16	5-18
Trajectory length	5	6	7	8	9
Entropy Coef.	1e-2	1e-3	1e-4	1e-5	1e-6
Learning rate	1e-5	1.5e-5	2e-5	2.5e-5	3e-5
PPO ϵ	.1	.15	.2	.25	.3
Num. PPO It.	200	200	200	200	200
Epoch/It.	10	10	10	10	10
Trajectory/It.	300	300	300	300	300

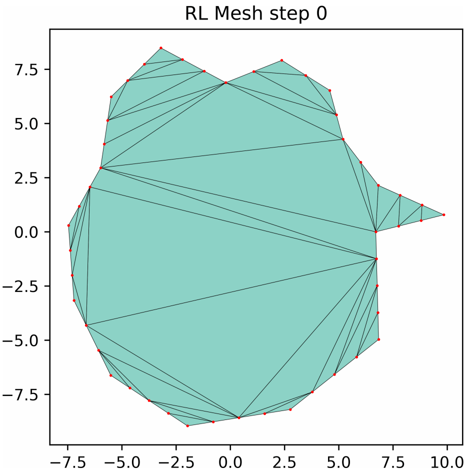
- 5 training phases
- Increasing domain size and complexity
- Adaptive learning rate and PPO ϵ
- Entropy coefficient decreases to stabilize policy

Model Size Experiment: Training Reward

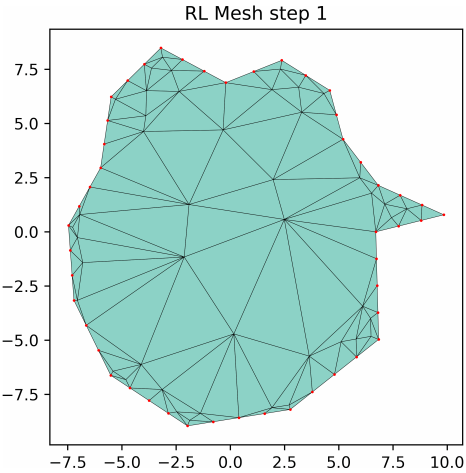


- Effect of GNN depth and width
- Best: $\ell = 2, d = 8$ and $\ell = 4, d = 16$
- Baseline: $\ell = 2, d = 8$
- Training does not always correlate with eval

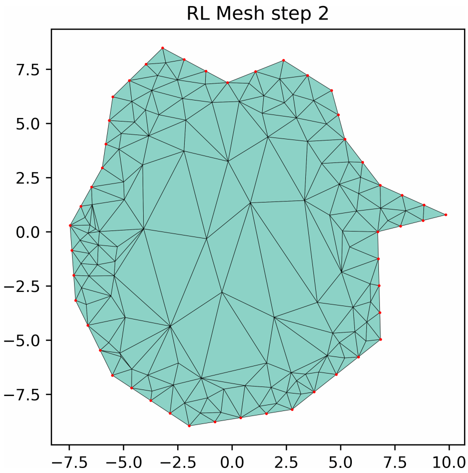
Examples



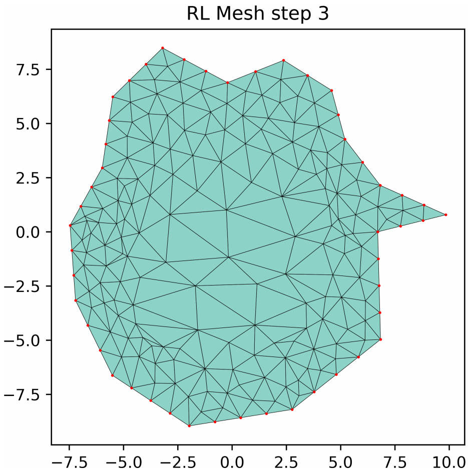
Examples



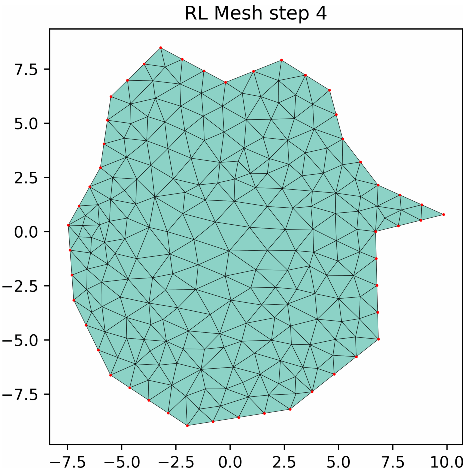
Examples



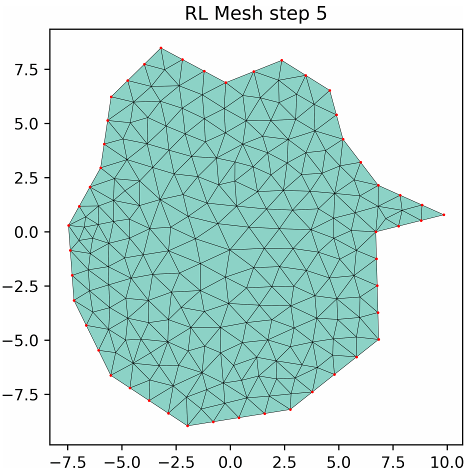
Examples



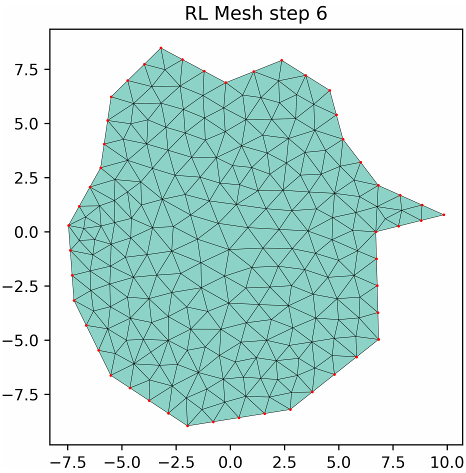
Examples



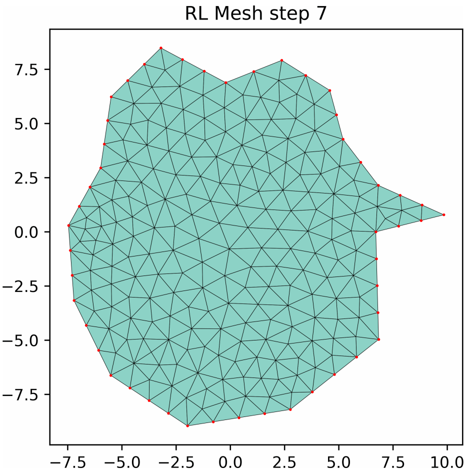
Examples



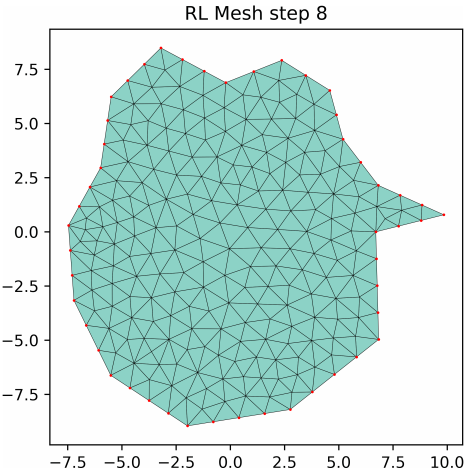
Examples



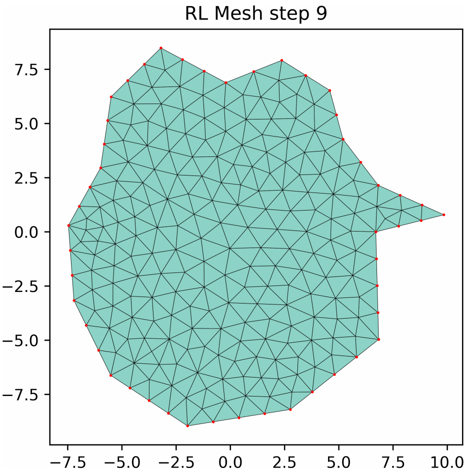
Examples



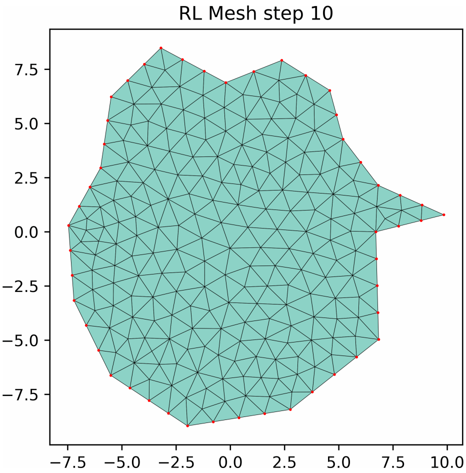
Examples



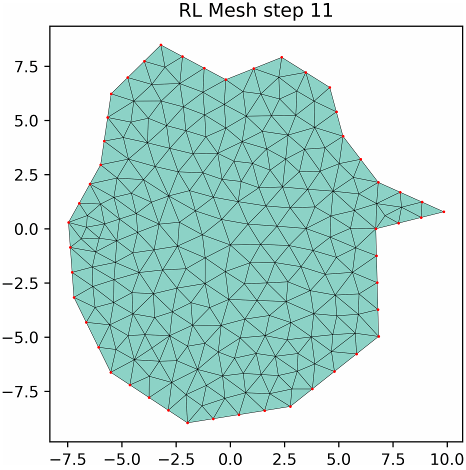
Examples



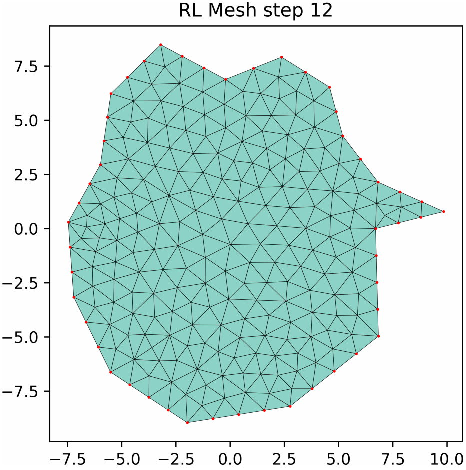
Examples



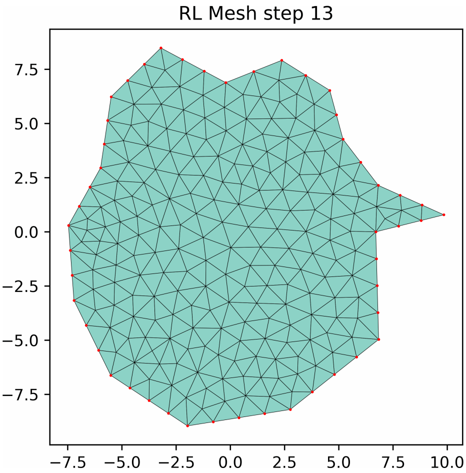
Examples



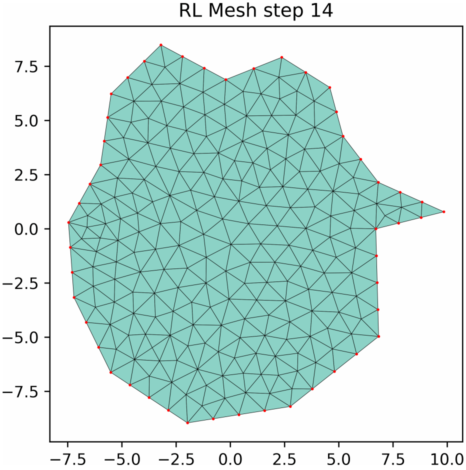
Examples



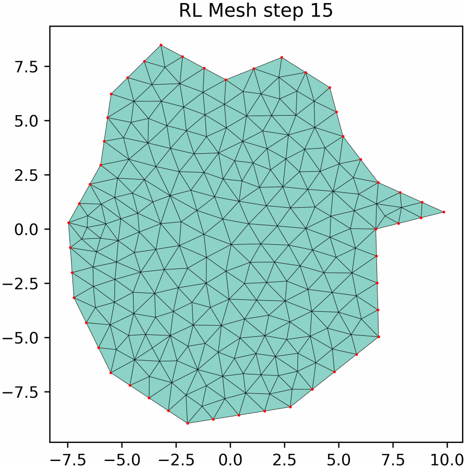
Examples



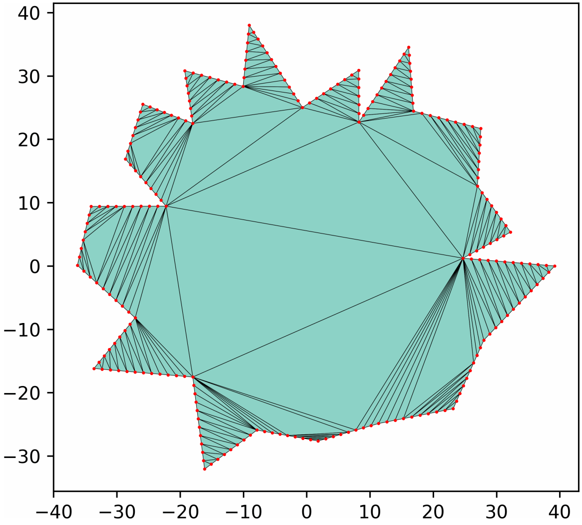
Examples



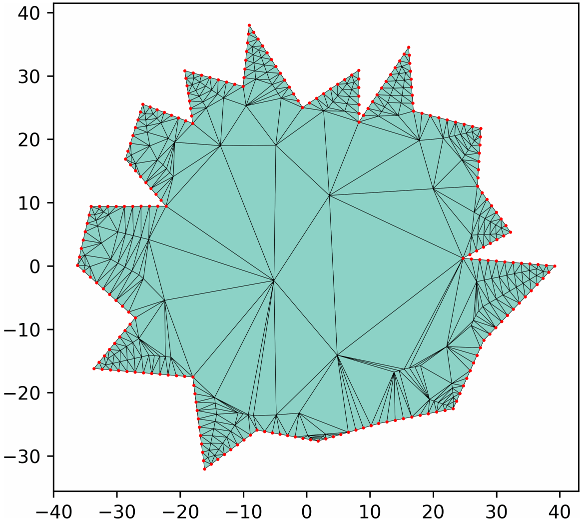
Examples



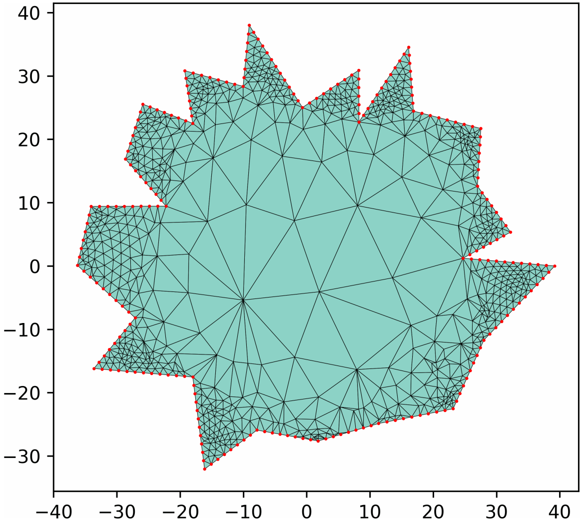
Examples



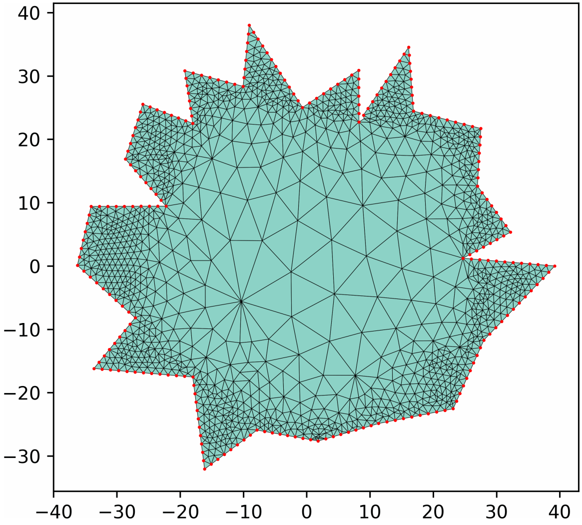
Examples



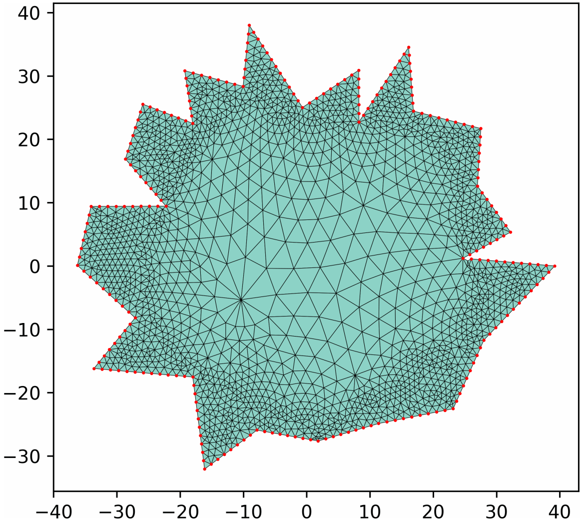
Examples



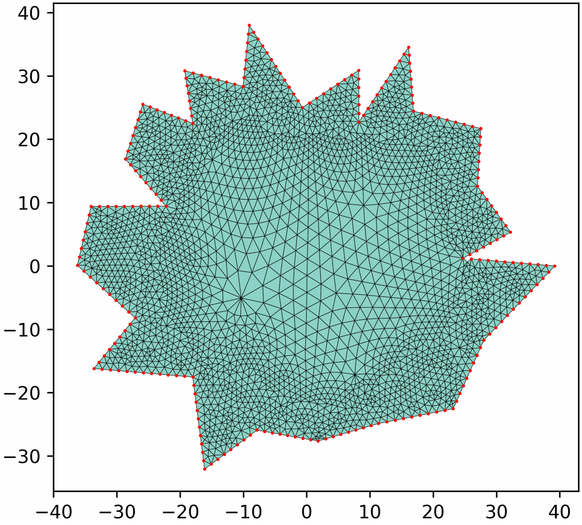
Examples



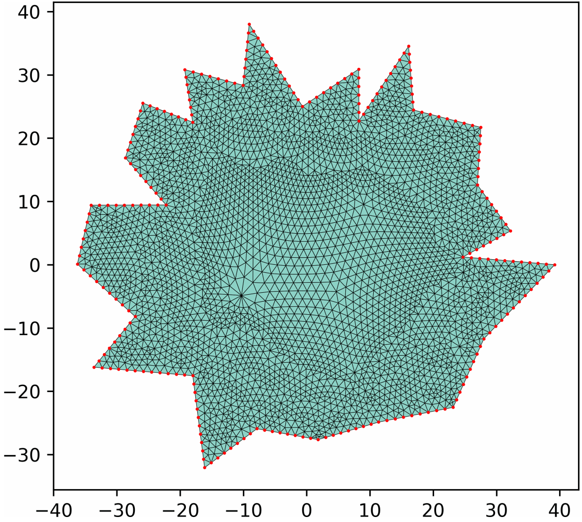
Examples



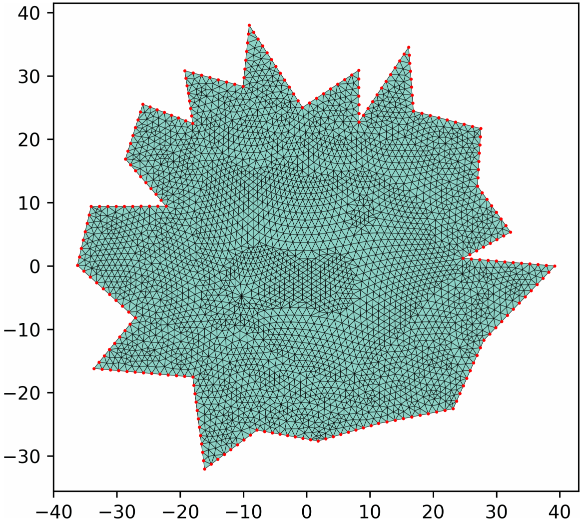
Examples



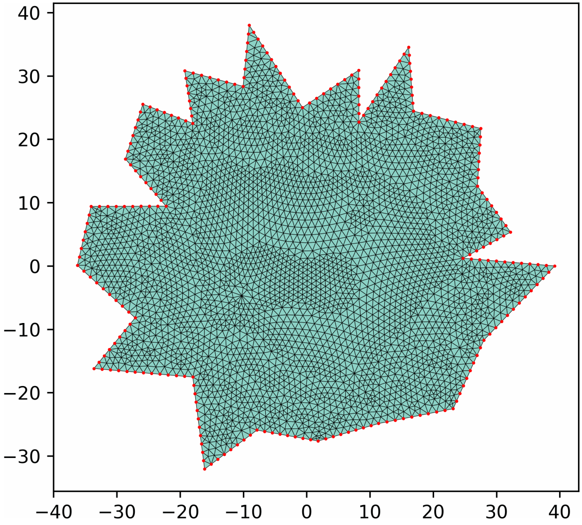
Examples



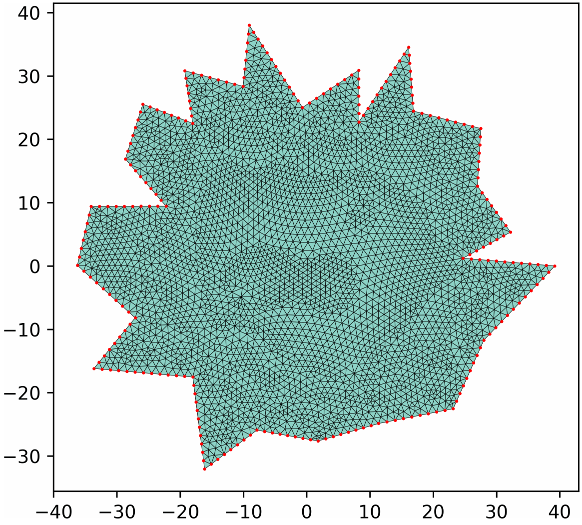
Examples



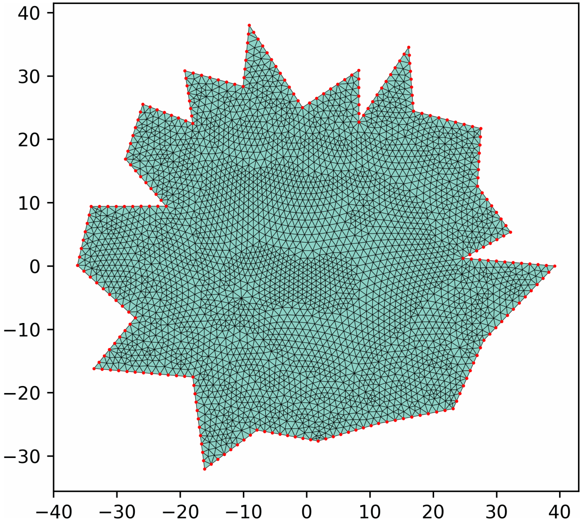
Examples



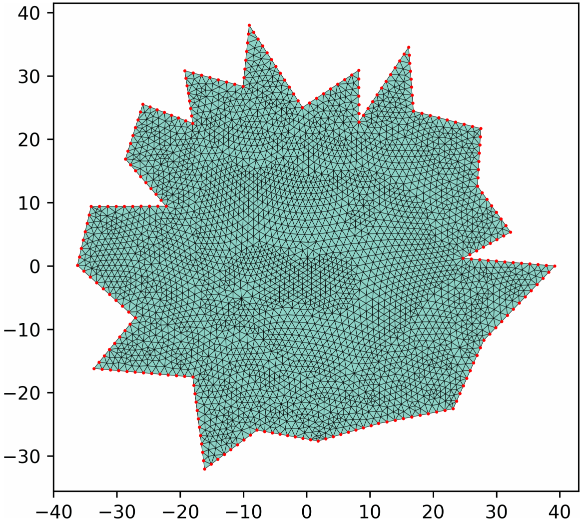
Examples



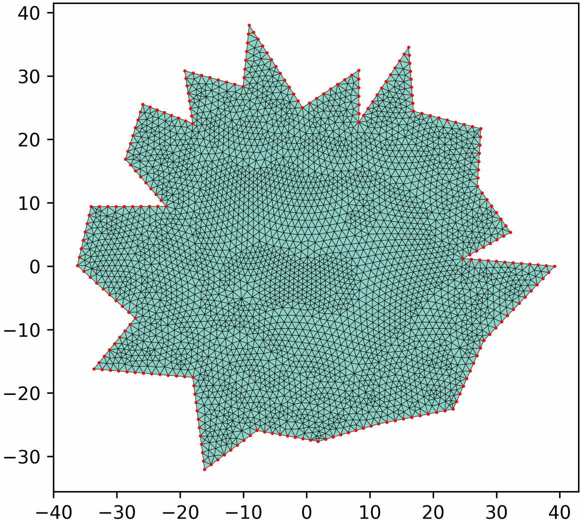
Examples



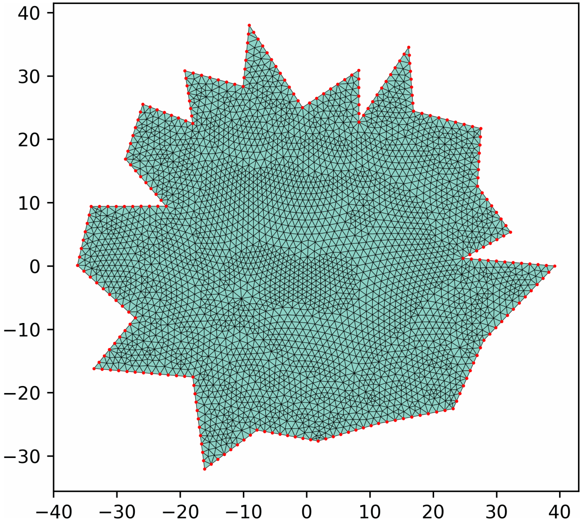
Examples



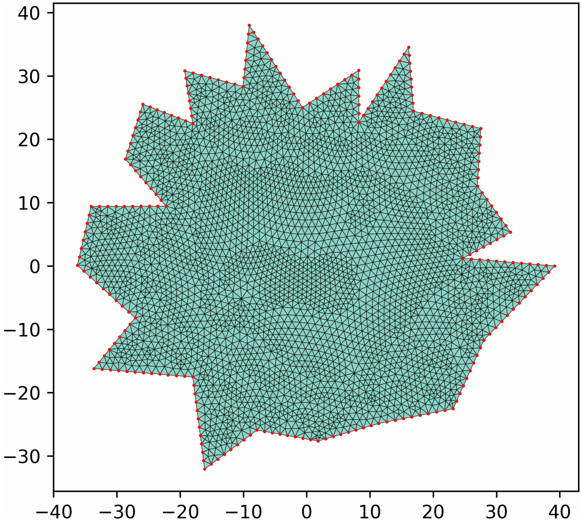
Examples



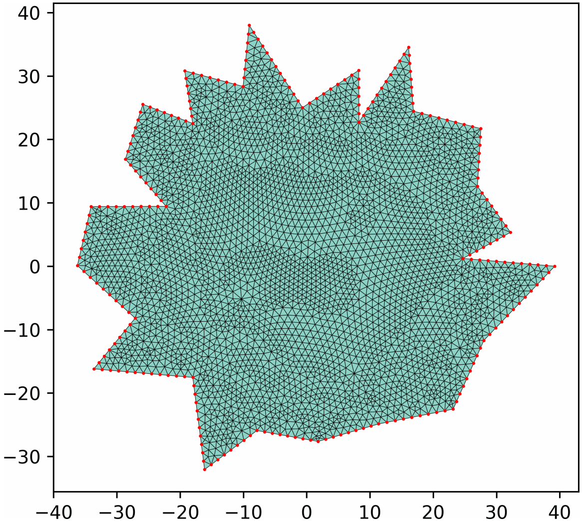
Examples



Examples



Examples



Model Size: Quality Metrics

d	ℓ	$|\theta|$	q_a (Mean)	q_a (Min)	q_a (SD)	q_e (Mean)	q_e (Min)	q_e (SD)	q_r (Mean)	q_r (Min)	q_r (SD)	q_v (Mean)	q_v (Min)	q_v (SD)
4	2	1107	0.841	0.111	0.133	0.582	-0.252	0.256	0.936	0.582	0.070	0.089	-1.976	0.632
8	2	3791	**0.860**	**0.198**	0.118	0.862	0.327	0.114	**0.948**	**0.619**	0.060	0.779	-0.127	0.182
16	2	13959	0.821	0.100	0.136	0.857	0.359	0.110	0.922	0.552	0.073	0.795	-0.067	0.161
32	2	53495	0.839	0.180	0.123	0.877	0.497	0.092	0.936	0.590	0.062	0.857	0.293	0.119
4	4	2029	0.836	0.102	0.131	0.853	0.404	0.111	0.932	0.544	0.071	0.765	0.079	0.1599
8	4	7033	0.824	0.041	0.136	0.873	0.425	0.098	0.924	0.494	0.075	0.821	0.338	0.130
16	4	26065	0.839	0.191	0.123	**0.883**	**0.519**	0.089	0.936	0.597	0.062	**0.865**	**0.388**	0.108
32	4	100225	0.826	0.101	0.133	0.873	0.366	0.102	0.926	0.556	0.070	0.842	0.028	0.144

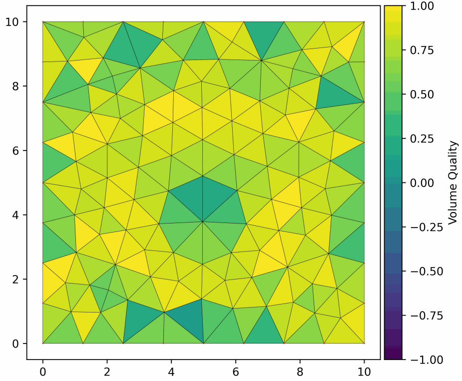
- Best q_e: $\ell = 4$, $d = 16$
- Best q_a: $\ell = 2$, $d = 8$
- Tradeoff: performance vs. size

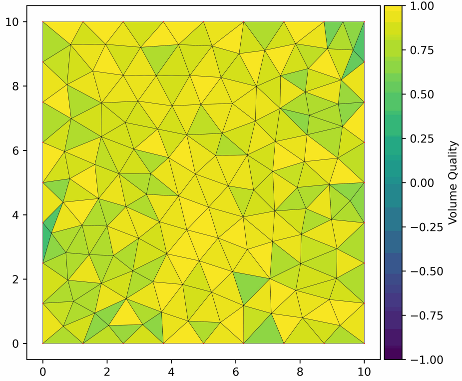
Reward Function Variation

Det.	w_a	w_r	w_e	w_v	q_a (Mean)	q_a (Min)	q_a (SD)	q_e (Mean)	q_e (Min)	q_e (SD)	q_r (Mean)	q_r (Min)	q_r (SD)	q_v (Mean)	q_v (Min)	q_v (SD)
Y	.5	0	.5	0	**0.860**	0.198	0.118	0.862	0.327	0.114	**0.948**	**0.619**	0.060	0.779	-0.127	0.182
N	.5	0	.5	0	0.841	0.043	0.129	0.847	0.378	0.113	0.936	0.484	0.070	0.751	0.137	0.170
Y	0	0	1	0	0.857	**0.202**	0.121	**0.868**	**0.394**	0.106	0.945	0.597	0.062	0.794	0.101	0.158
N	0	0	1	0	0.846	0.045	0.127	0.855	0.382	0.112	0.939	0.485	0.069	0.767	0.111	0.167
Y	0	0	0	1	0.827	0.140	0.132	0.765	0.082	0.166	0.928	0.605	0.068	0.588	-0.841	0.291
N	0	0	0	1	0.827	-0.226	0.139	0.877	0.331	0.097	0.925	0.261	0.085	**0.864**	**0.225**	0.116
Y	.25	.25	.25	.25	0.823	0.162	0.134	0.690	0.063	0.195	0.924	0.609	0.070	0.401	-0.897	0.350
N	.25	.25	.25	.25	0.836	-0.020	0.129	0.846	0.367	0.115	0.934	0.451	0.070	0.776	-0.012	0.173

- Reward: $w_a q_a + w_e q_e + w_r q_r + w_v q_v$
- Tailored reward improves specific metrics
- Stochastic policies can outperform deterministic

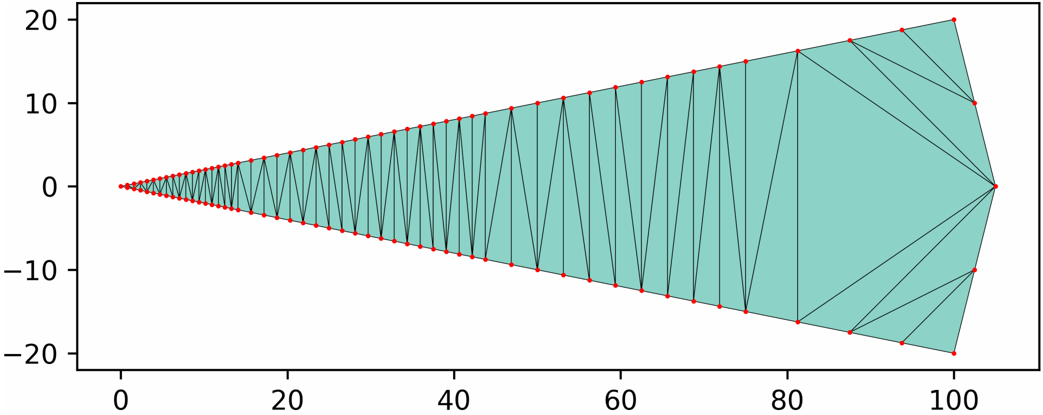
Visual Effect of Reward Variation



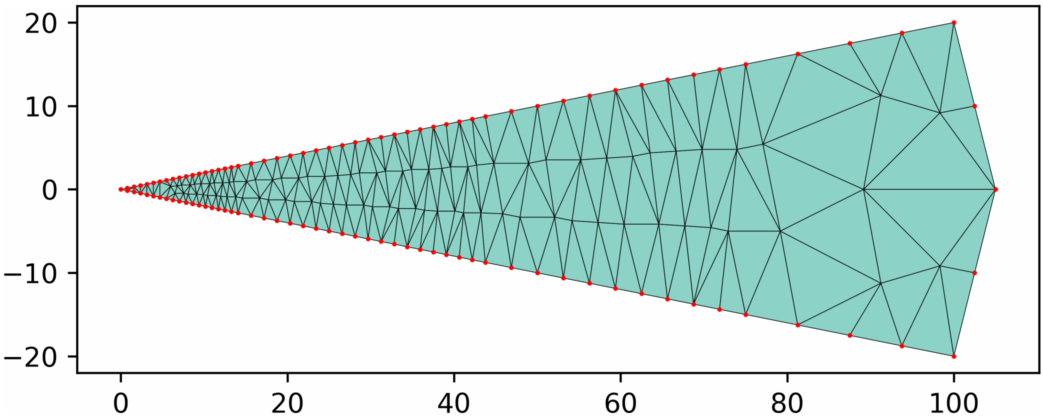


- Left: $w_a = w_e = 0.5$, deterministic
- Right: $w_v = 1$, stochastic
- Volume uniformity vs. shape quality

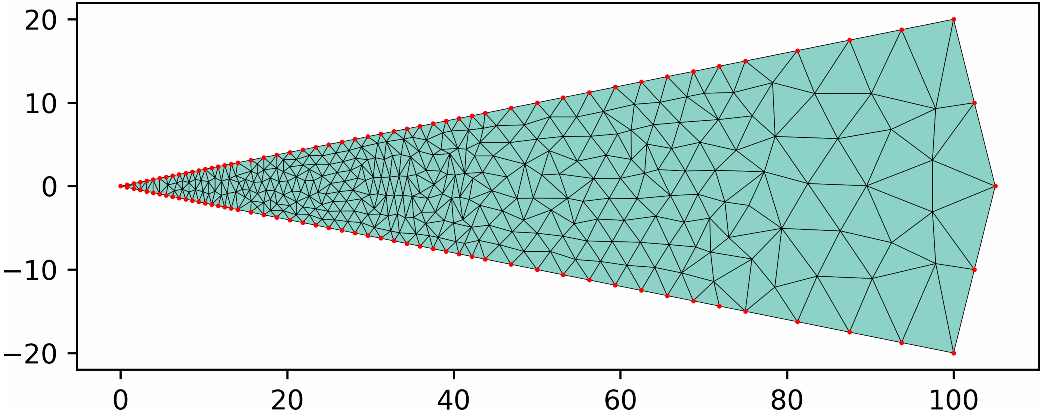
Example: Size function



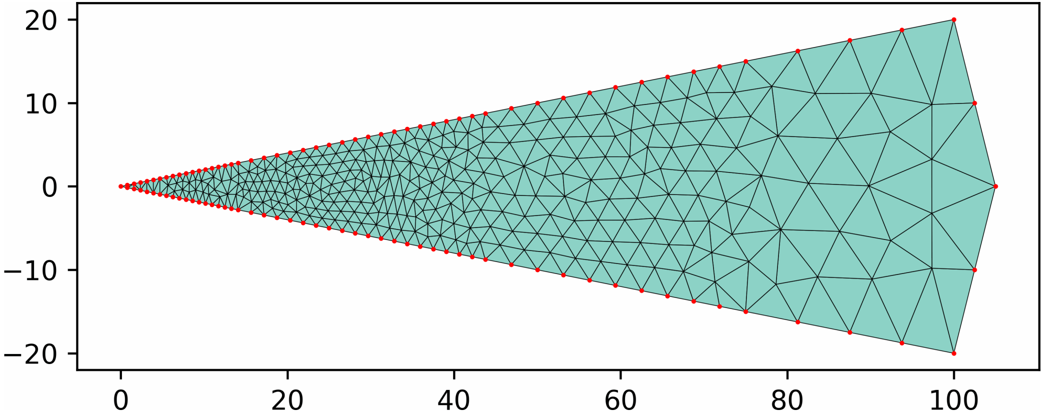
Example: Size function



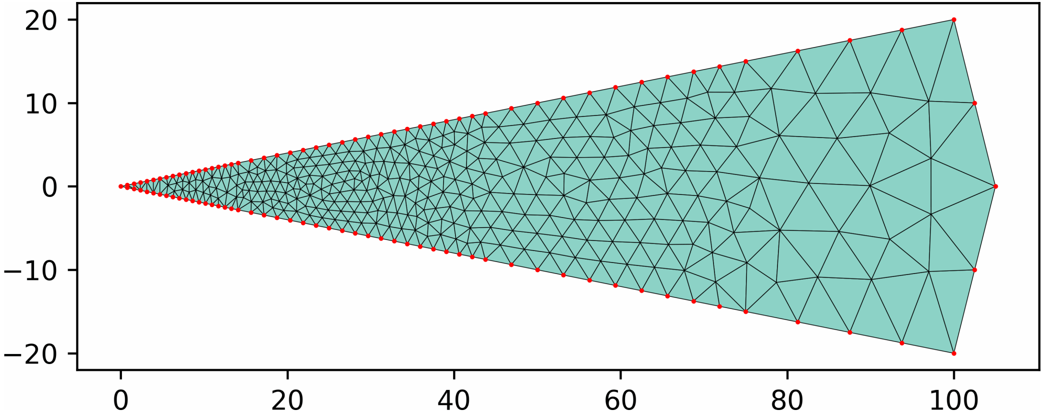
Example: Size function



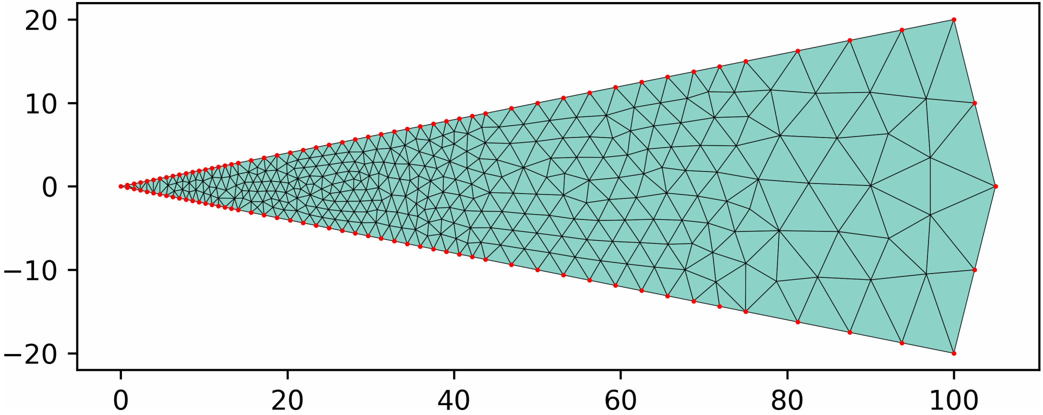
Example: Size function



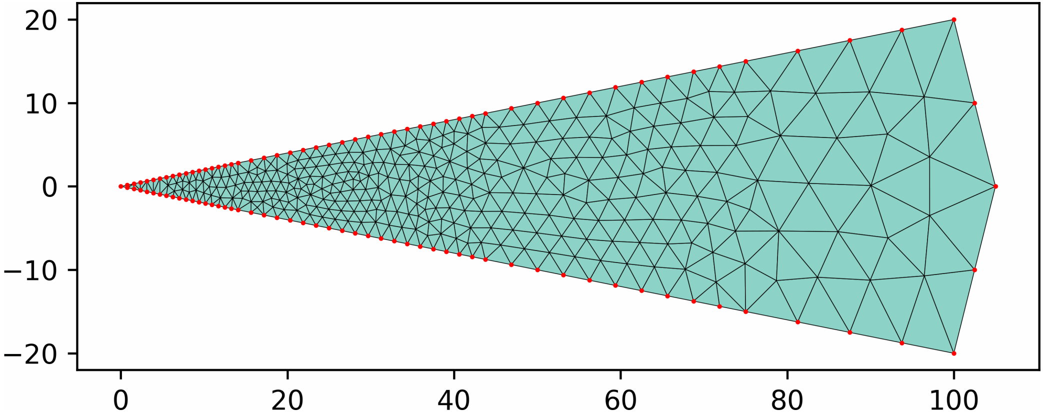
Example: Size function



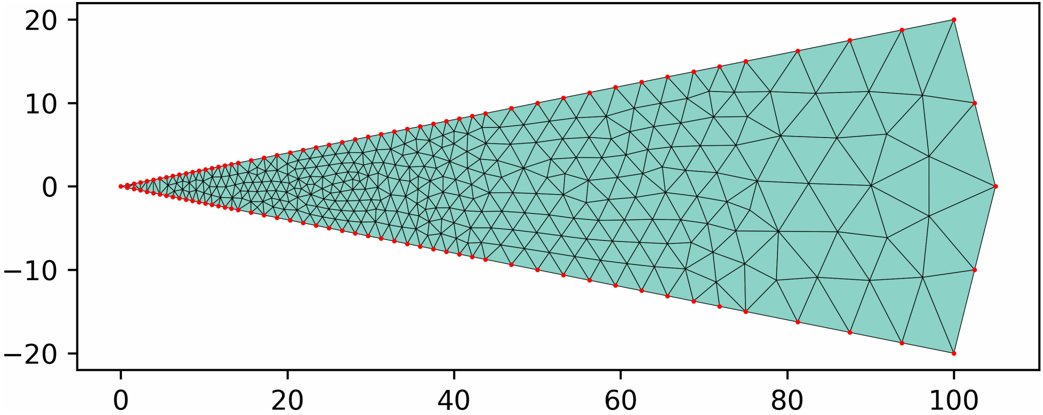
Example: Size function



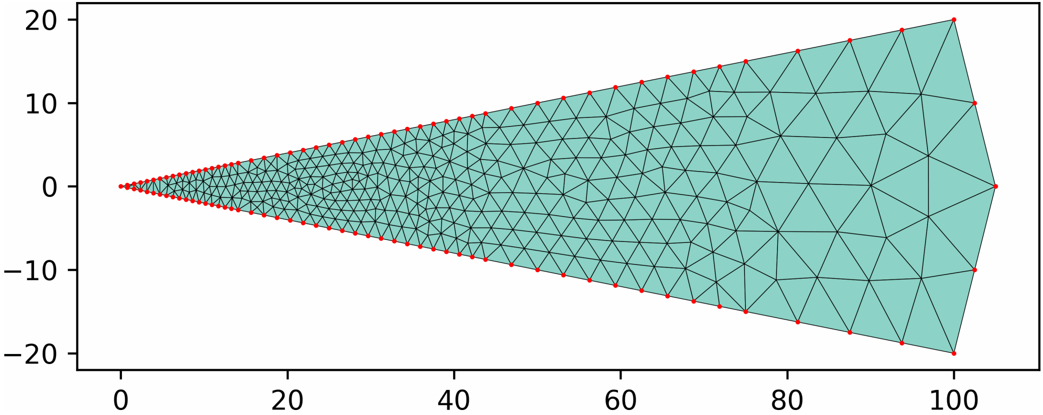
Example: Size function



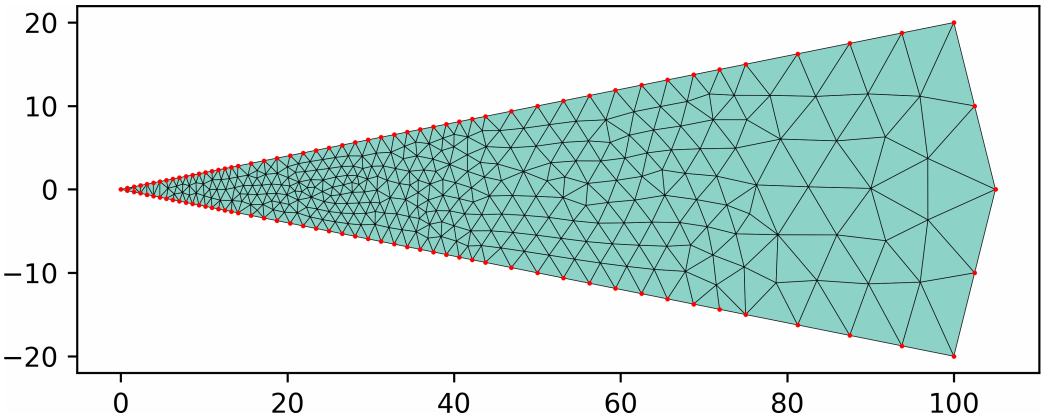
Example: Size function



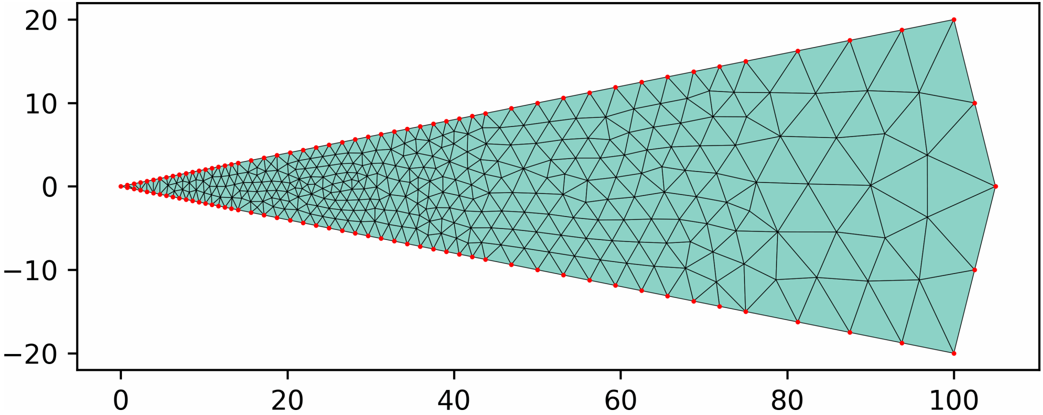
Example: Size function



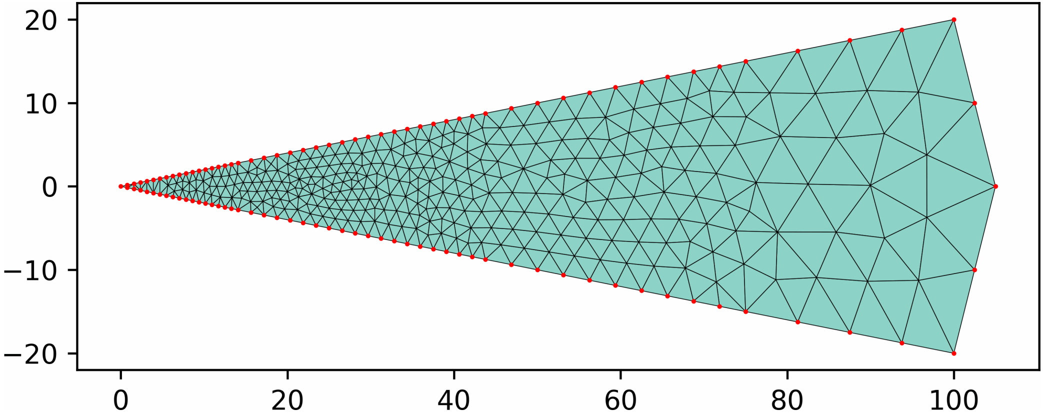
Example: Size function



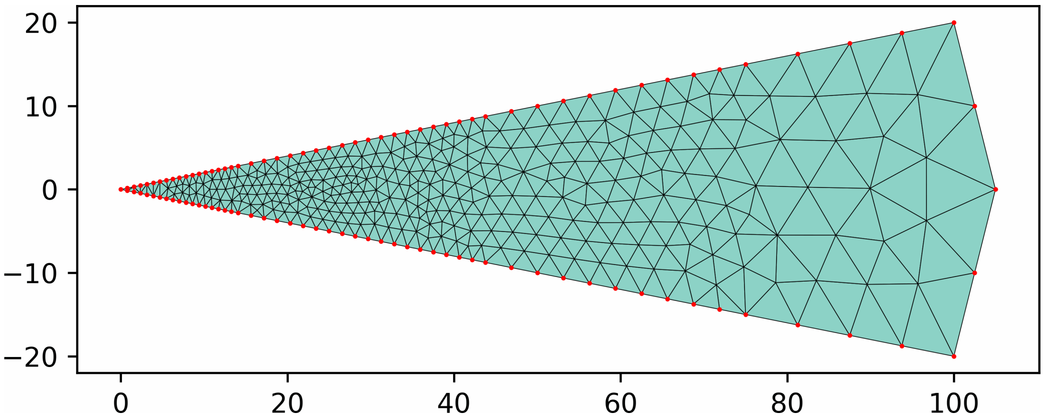
Example: Size function



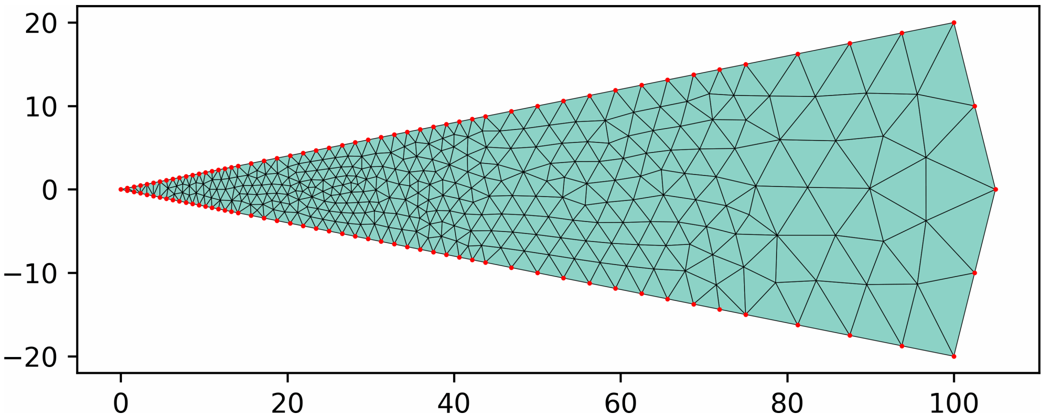
Example: Size function



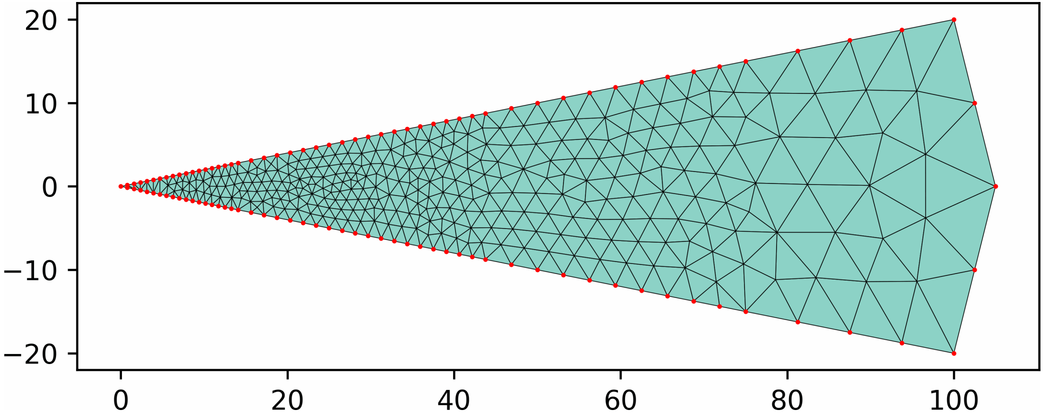
Example: Size function



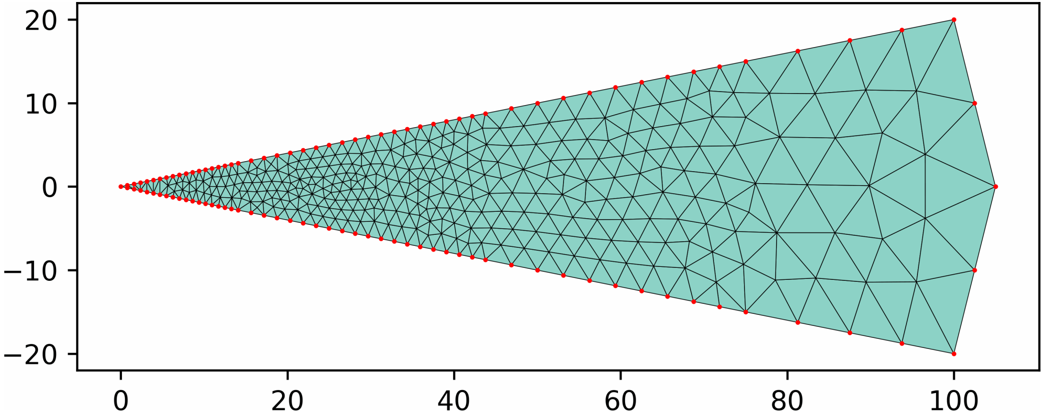
Example: Size function



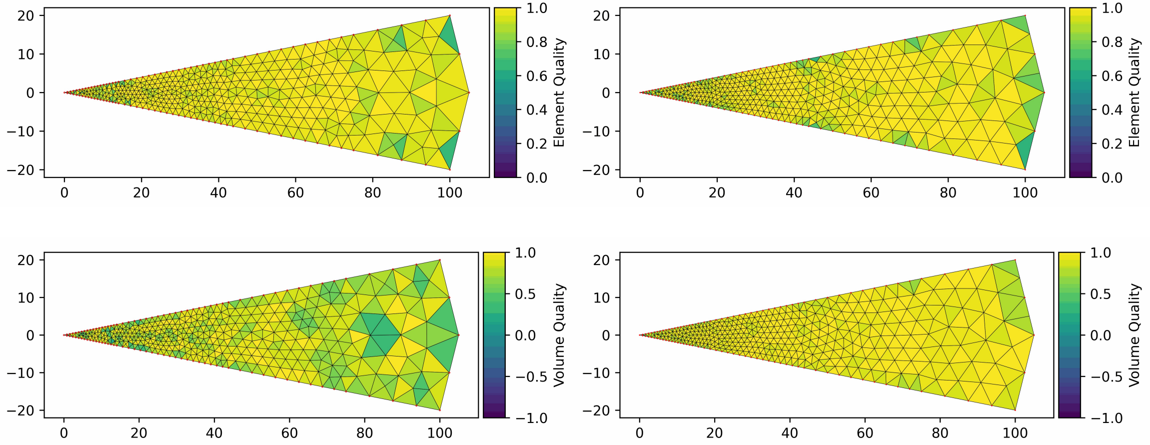
Example: Size function



Example: Size function



Mesh Generation with Size Function

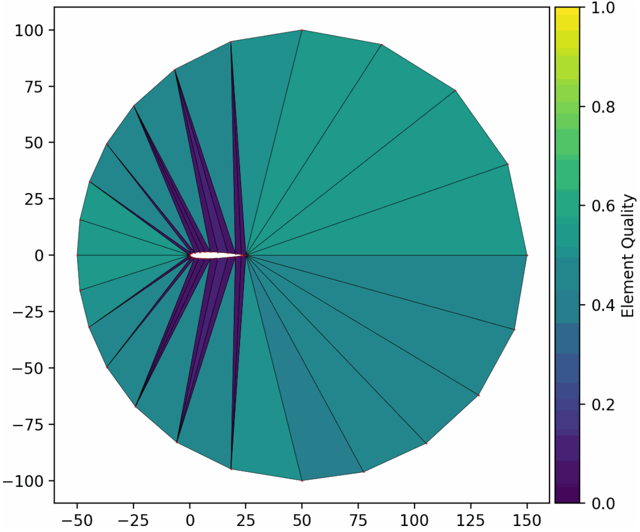


Proposed method

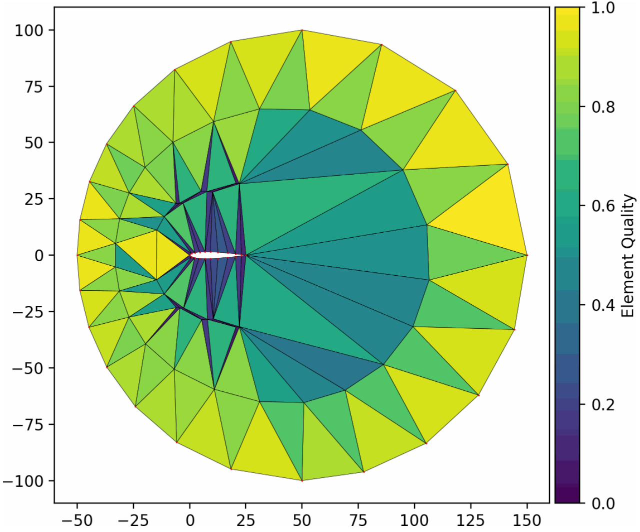
DistMesh for comparison

- Trained on constant size, but generalizes to variable $h(x)$
- DistMesh outperforms on complex geometries

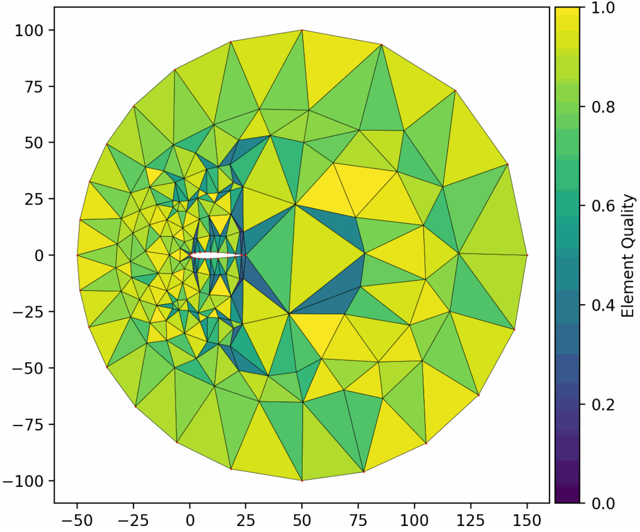
Example: NACA Airfoil Mesh



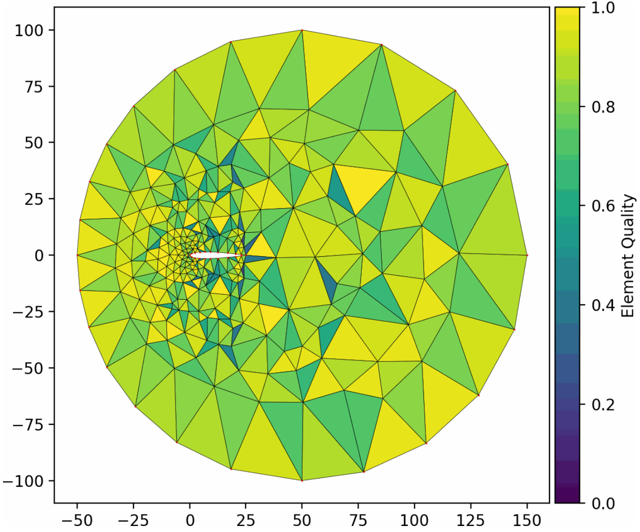
Example: NACA Airfoil Mesh



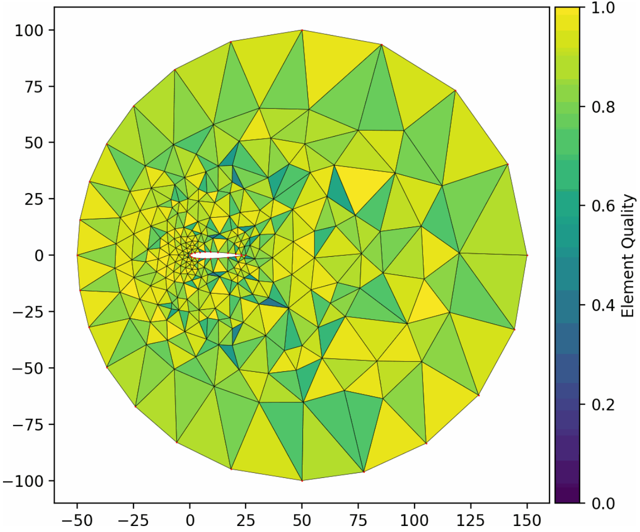
Example: NACA Airfoil Mesh



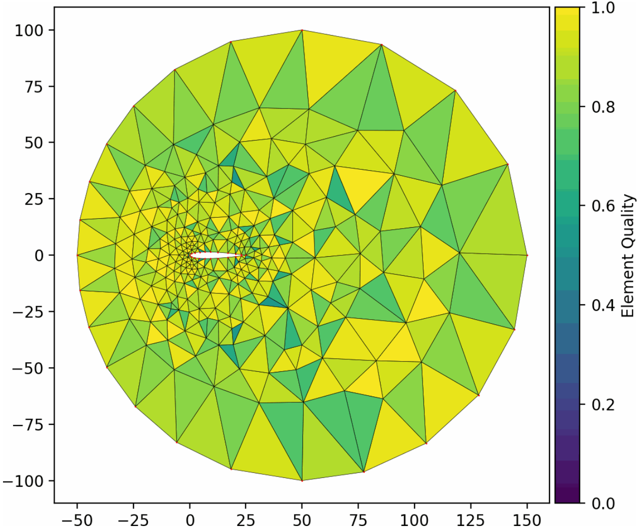
Example: NACA Airfoil Mesh



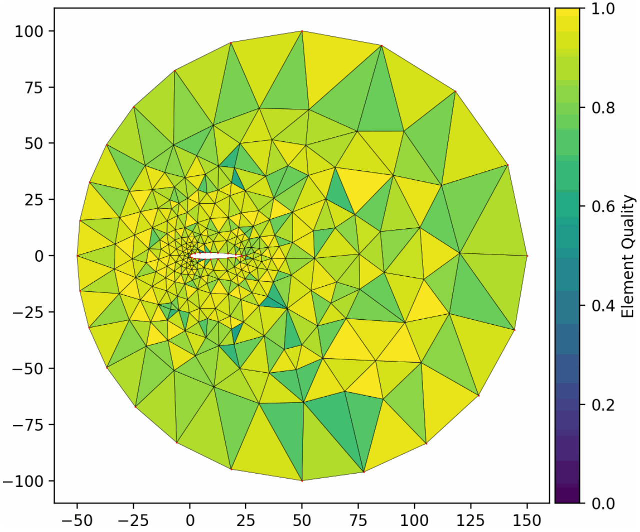
Example: NACA Airfoil Mesh



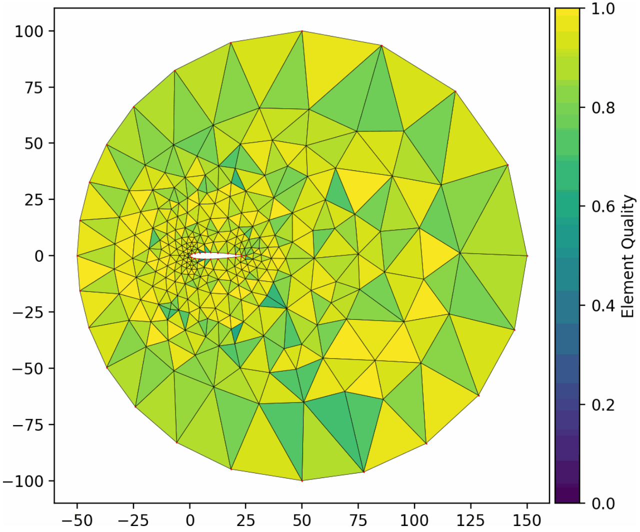
Example: NACA Airfoil Mesh



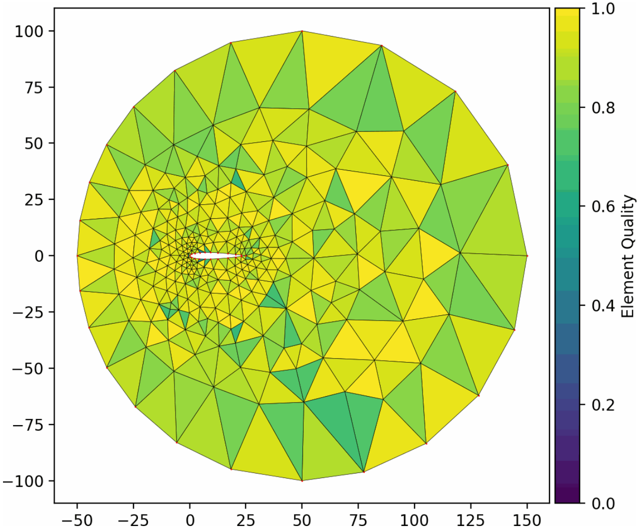
Example: NACA Airfoil Mesh



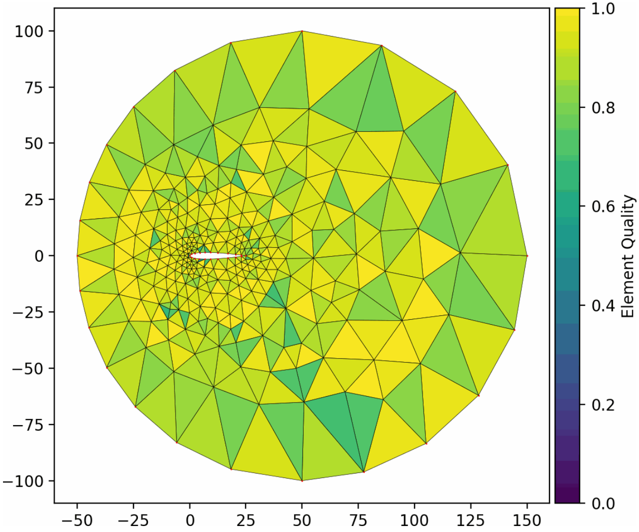
Example: NACA Airfoil Mesh



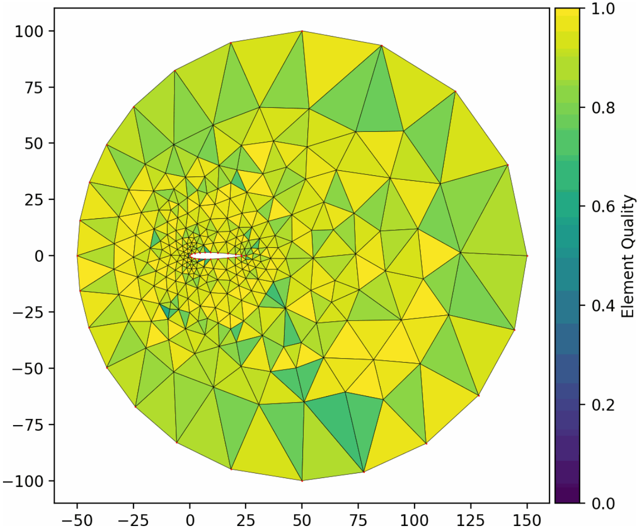
Example: NACA Airfoil Mesh



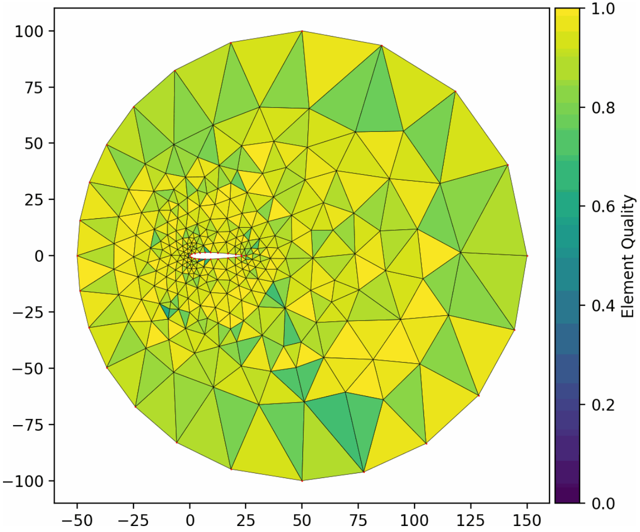
Example: NACA Airfoil Mesh



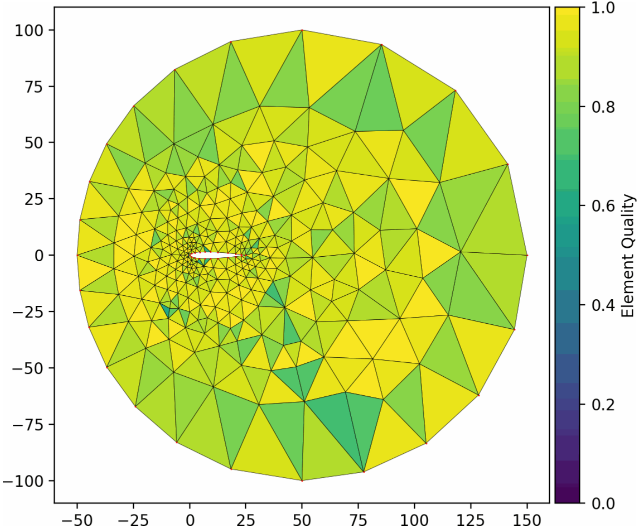
Example: NACA Airfoil Mesh



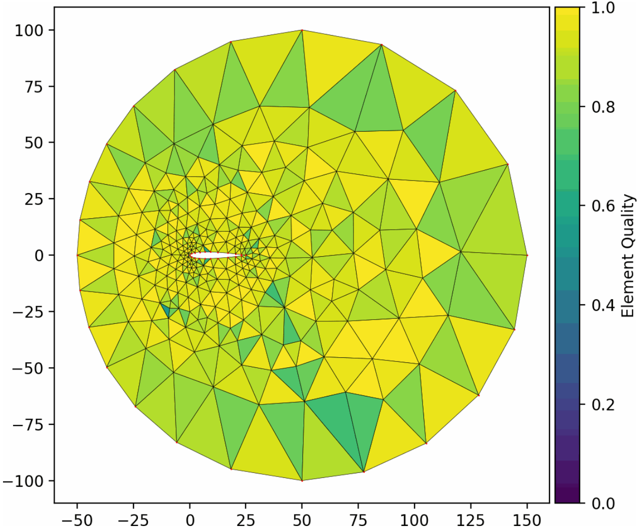
Example: NACA Airfoil Mesh



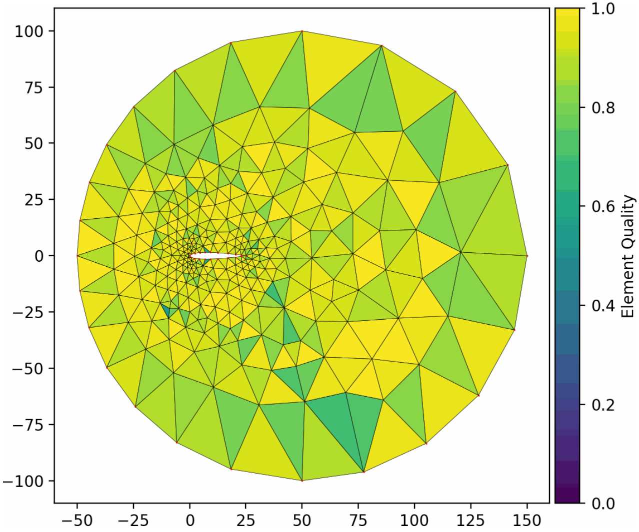
Example: NACA Airfoil Mesh



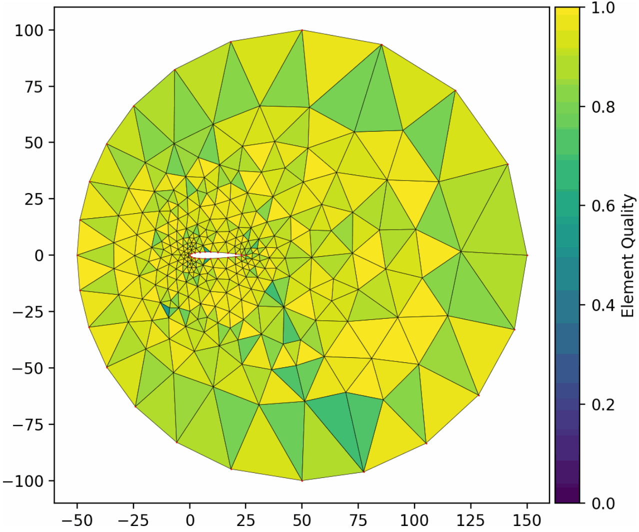
Example: NACA Airfoil Mesh



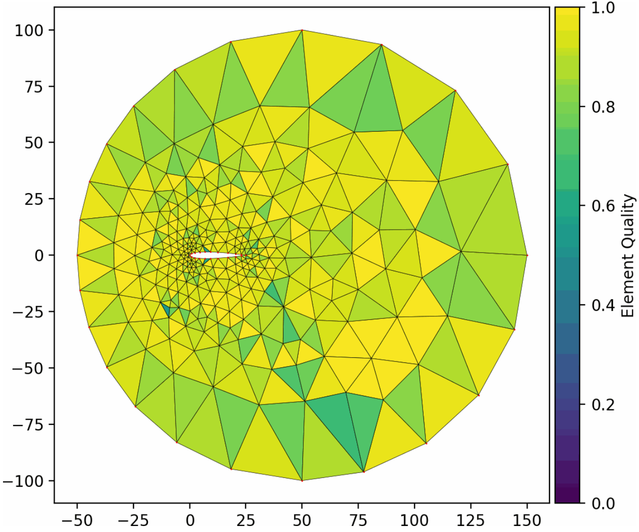
Example: NACA Airfoil Mesh



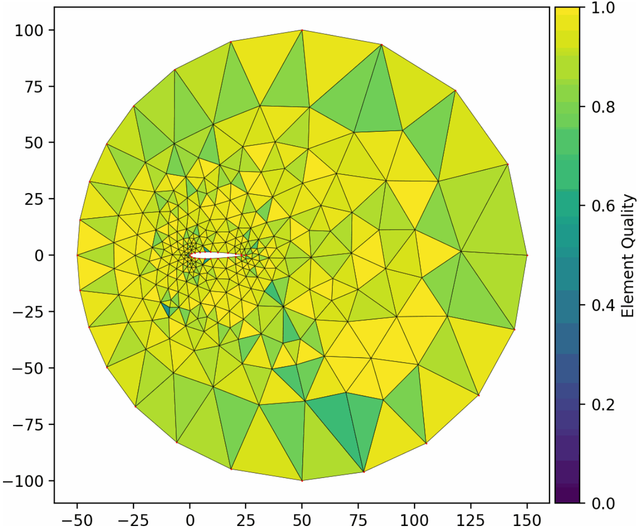
Example: NACA Airfoil Mesh



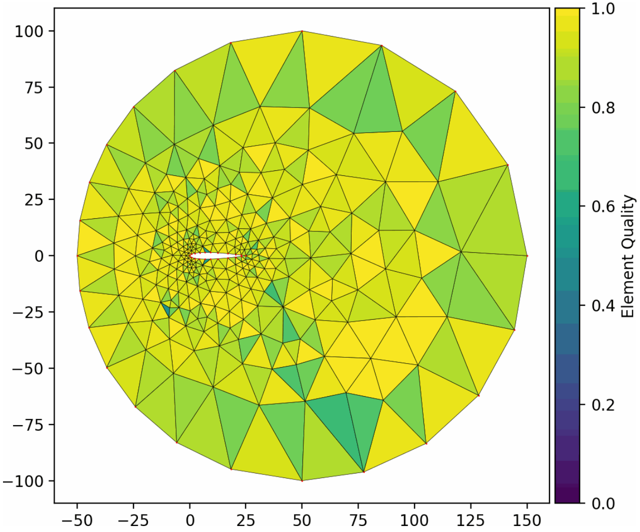
Example: NACA Airfoil Mesh



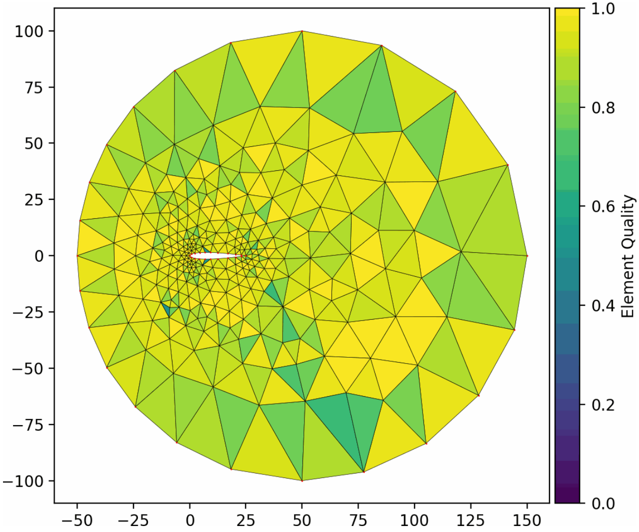
Example: NACA Airfoil Mesh



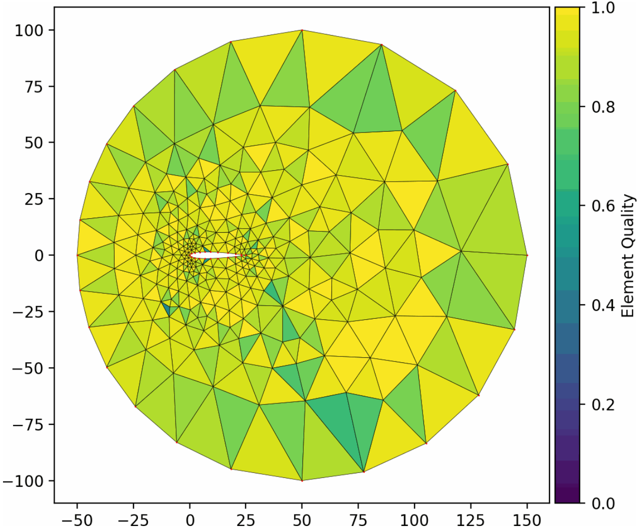
Example: NACA Airfoil Mesh



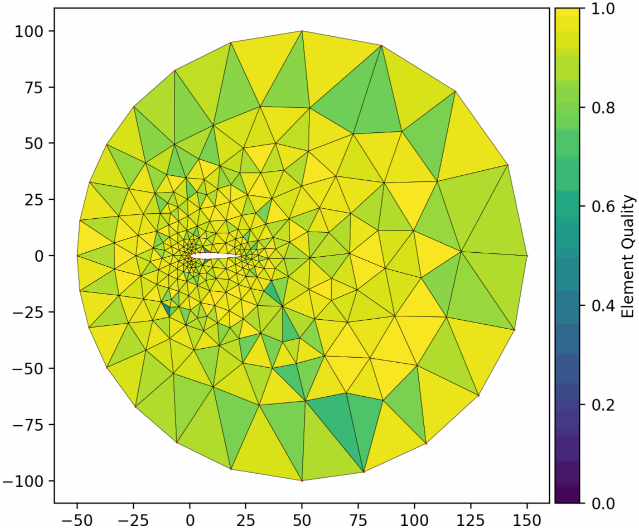
Example: NACA Airfoil Mesh



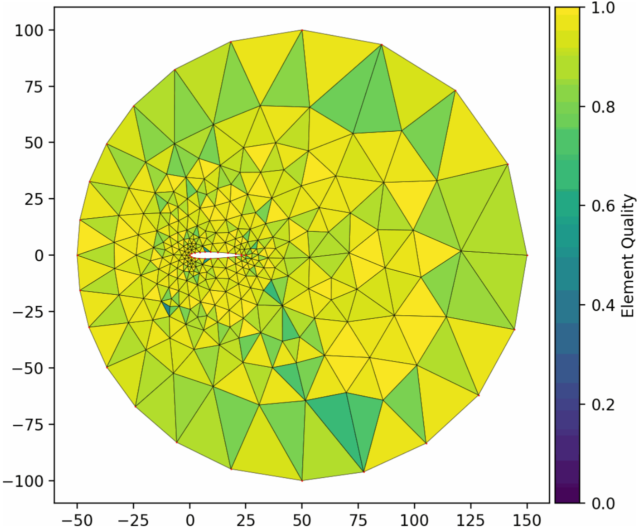
Example: NACA Airfoil Mesh



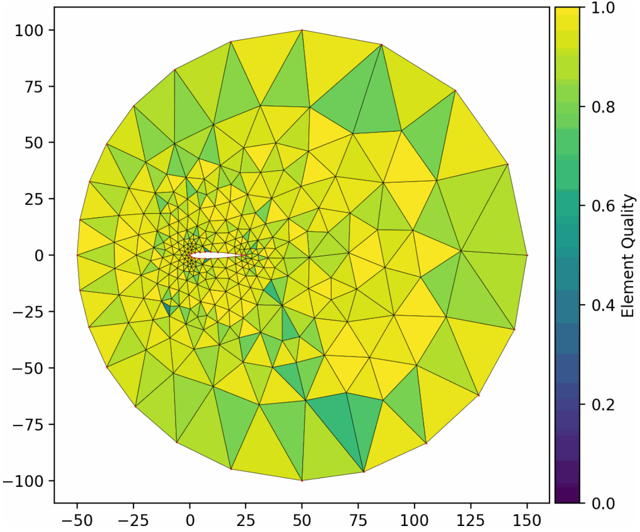
Example: NACA Airfoil Mesh



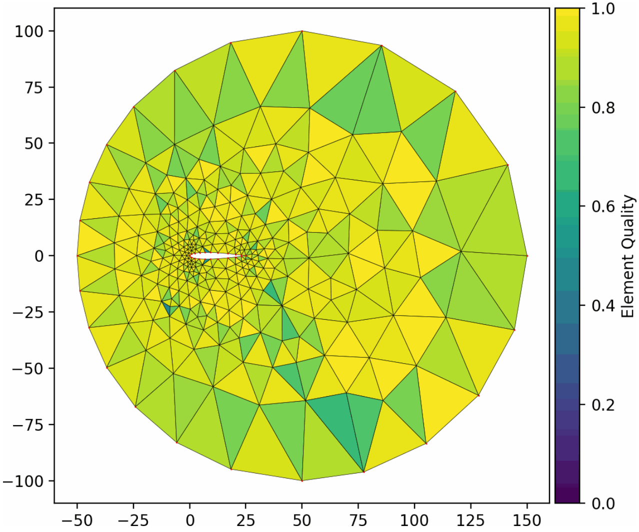
Example: NACA Airfoil Mesh



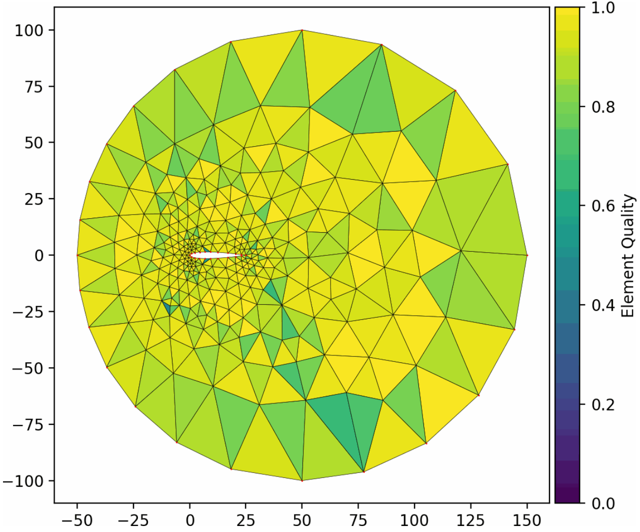
Example: NACA Airfoil Mesh



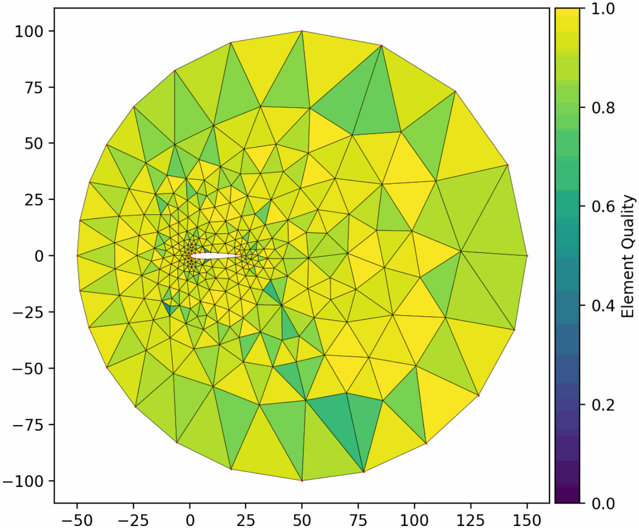
Example: NACA Airfoil Mesh



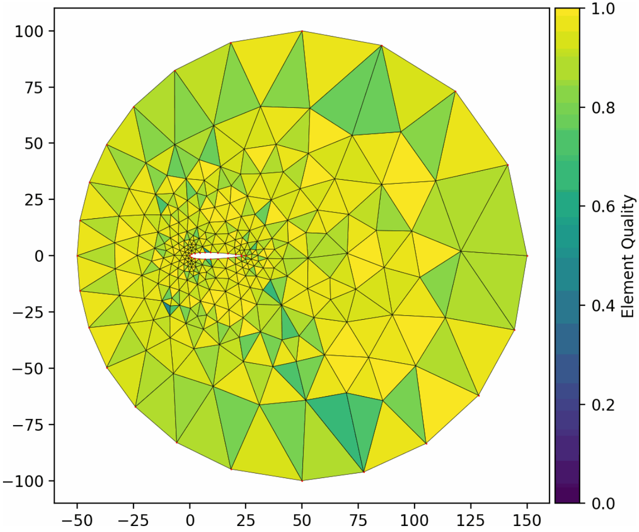
Example: NACA Airfoil Mesh



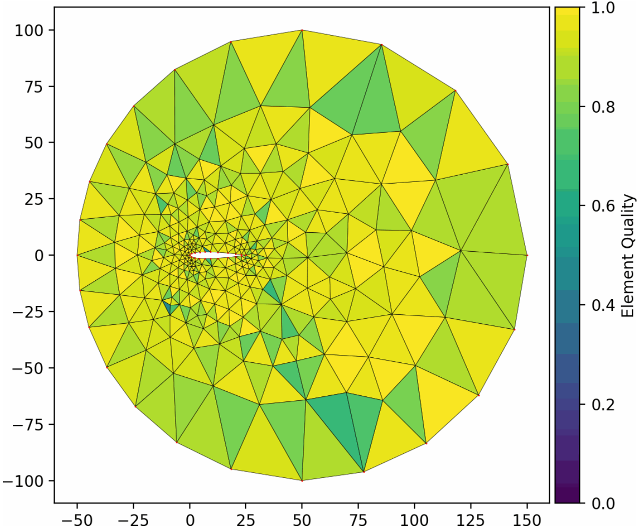
Example: NACA Airfoil Mesh



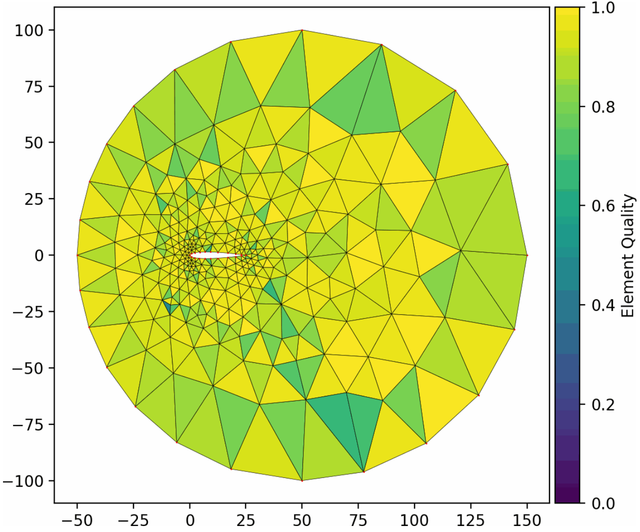
Example: NACA Airfoil Mesh



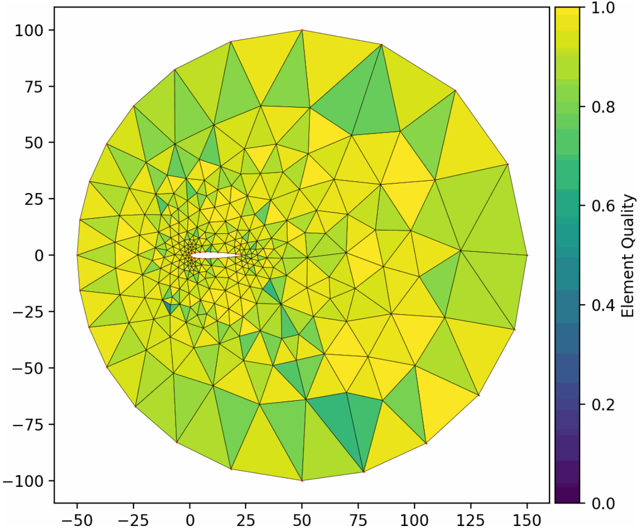
Example: NACA Airfoil Mesh



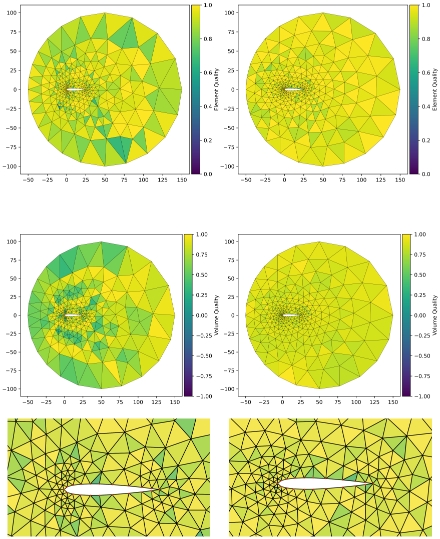
Example: NACA Airfoil Mesh



Example: NACA Airfoil Mesh



NACA Airfoil Mesh



New method DistMesh

- Complex curved geometry with sharp trailing edge
- RL-generated mesh: mostly acceptable, room for refinement
- DistMesh achieves higher quality around boundary

Conclusions and Future Directions

Summary of Contributions

- **New Paradigm:** Formulated meshing as a sequential “game” (RL).
- **Topology (Part I):** Learned optimal connectivity for Tri/Quad meshes.
- **Geometry (Part II):** Learned node placement with Delaunay constraints.
- **Result:** Heuristic-free agents that rival classical algorithms.

Future Work

- **3D Generation:** Extension to tetrahedral and hexahedral elements.
- **Unified Policy:** Learning topology and geometry simultaneously.
- **Advanced RL:** Integrating Monte Carlo Tree Search (MCTS).
- **Complexity:** Improving performance on variable-resolution domains.

References

[1] Narayanan, Pan, Persson. *Learning topological operations on meshes with application to block decomposition of polygons*. Computer-Aided Design, Vol. 175, pp. 103744 (2024) and arXiv:2309.06484.

[2] Narayanan, *Machine Learning Methods to Optimize the Geometry and Topology of Meshes*. Ph.D. thesis, University of California, Berkeley, May 2024.

[3] W. Thacher, P.-O. Persson, and Y. Pan, *Optimization of a Triangular Delaunay Mesh Generator using Reinforcement Learning*. Computer-Aided Design, Vol. 189, pp. 103964 (2025) and arXiv:2504.03610.