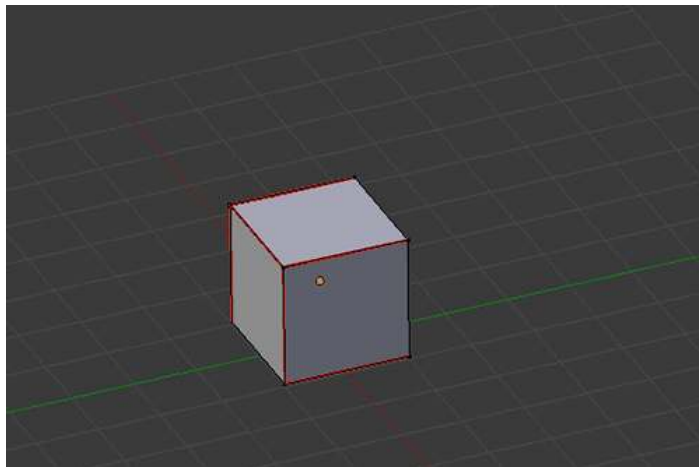


Introduction to Mesh Parameterization

Richard Liu



[Source](#)

Self Intro

5th (?) year PhD at UChicago 3DL lab



Self Intro

5th (?) year PhD at UChicago 3DL lab



3rd year presenting for SGI 🎉

Self Intro

5th (?) year PhD at UChicago 3DL lab



3rd year presenting for SGI 🎉

AI for mesh stuff (generation, animation, texturing, deformation, etc...)

Self Intro

5th (?) year PhD at UChicago 3DL lab



3rd year presenting for SGI 🎉

AI for mesh stuff (generation, animation, texturing, deformation, etc...)

A PhD journey through hair

2022



2023



2024



2025



Yesterday



Session Content

Definition and mathematical foundations

Computing fixed-boundary parameterizations

How to measure parameterization distortion

Jupyter notebook tutorial

Exercises: [101_parameterization_fundamentals](#),
[102_parameterization_exercises](#)

Source Material

Hormann, K., Polthier, K., Sheffer, A. “Mesh Parameterization: Theory and Practice”. Siggraph Asia 2008 Course. <https://www.cs.princeton.edu/courses/archive/fall12/cos526/papers/hormann08.pdf>

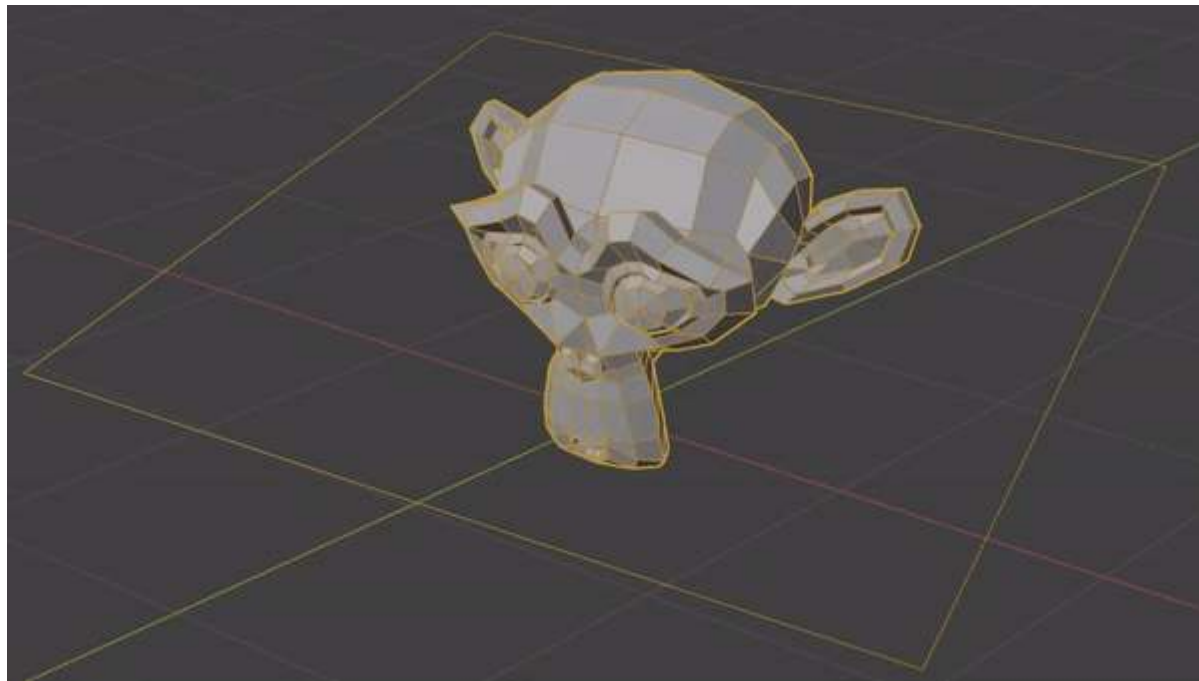
Jacobson, A. Geometry processing course parameterization assignment. <https://github.com/alecjacobson/geometry-processing-parameterization>

“Parameterization I”. Stanford CS-468. https://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/12_Parameterization1.pdf

Sheffer, A., Praun, E., Rose, K. “Mesh Parameterization Methods and Their Applications”. Trends in CG and Vision 2006. <https://people.eecs.berkeley.edu/~jrs/meshpapers/ShefferPraunRose.pdf>

Part 1: Foundations

What is mesh parameterization?

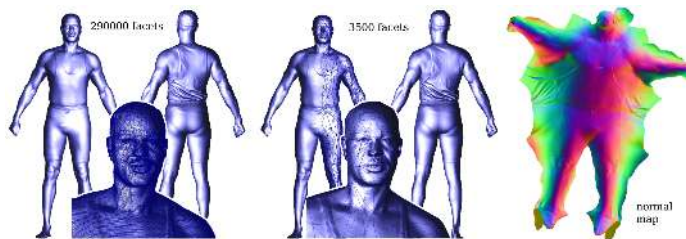


**Process of mapping
3D surface mesh onto
2D plane**

[Source](#)

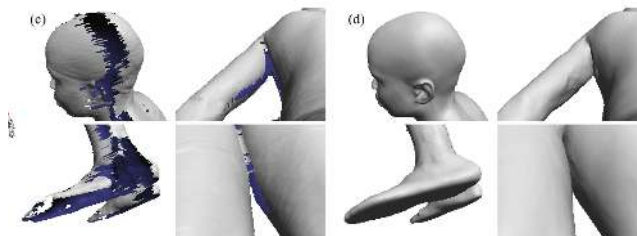
Lots of applications ...

Normal Mapping/Simplification



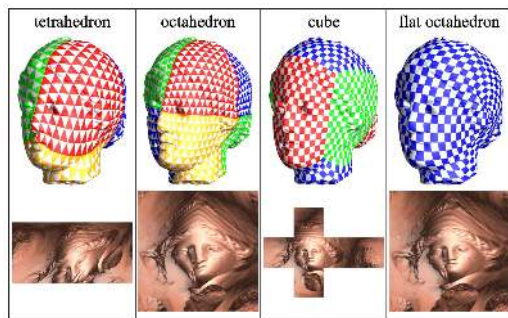
[Source](#)

Shape Completion



[Source](#)

Remeshing



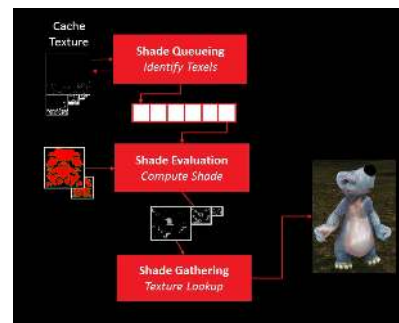
[Source](#)

Detail Transfer



[Source](#)

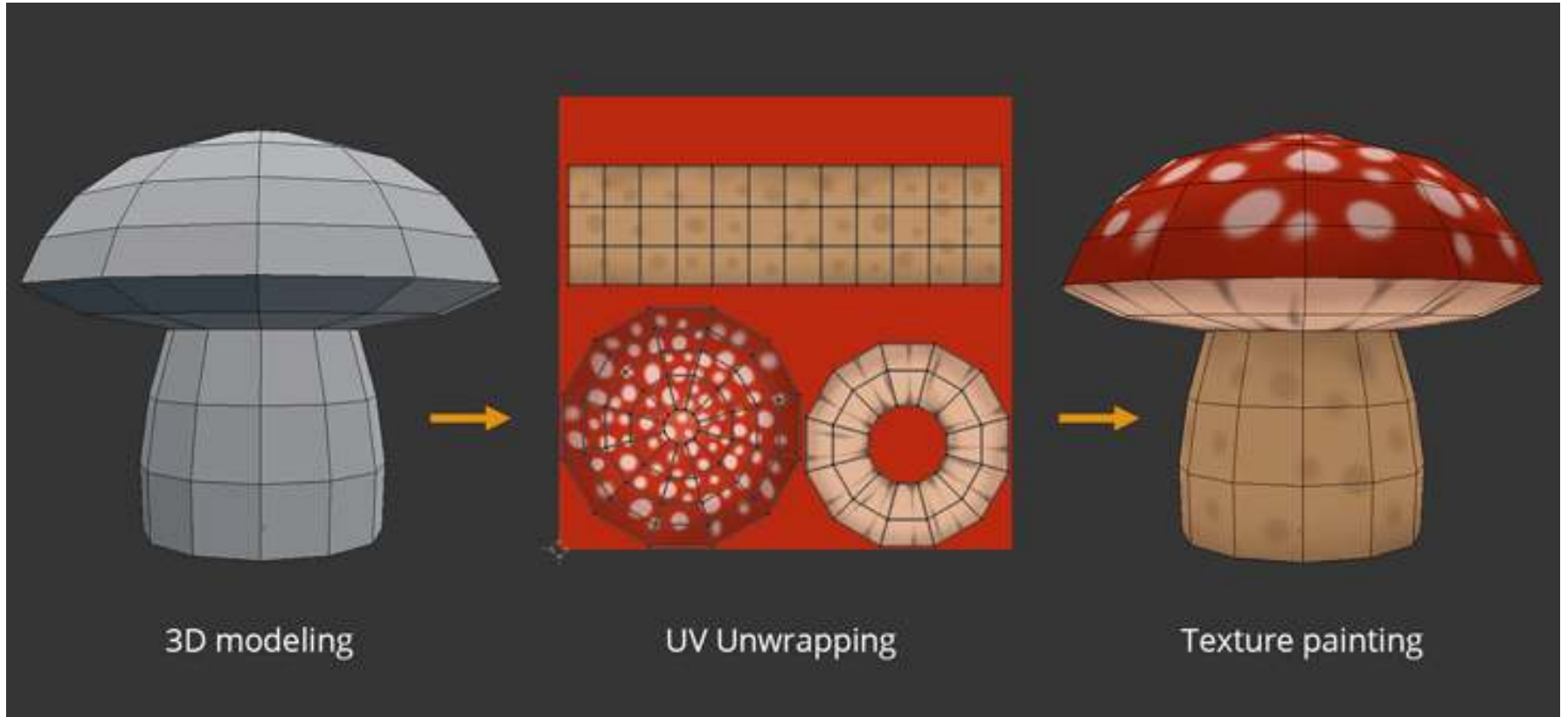
Texture Space Shading



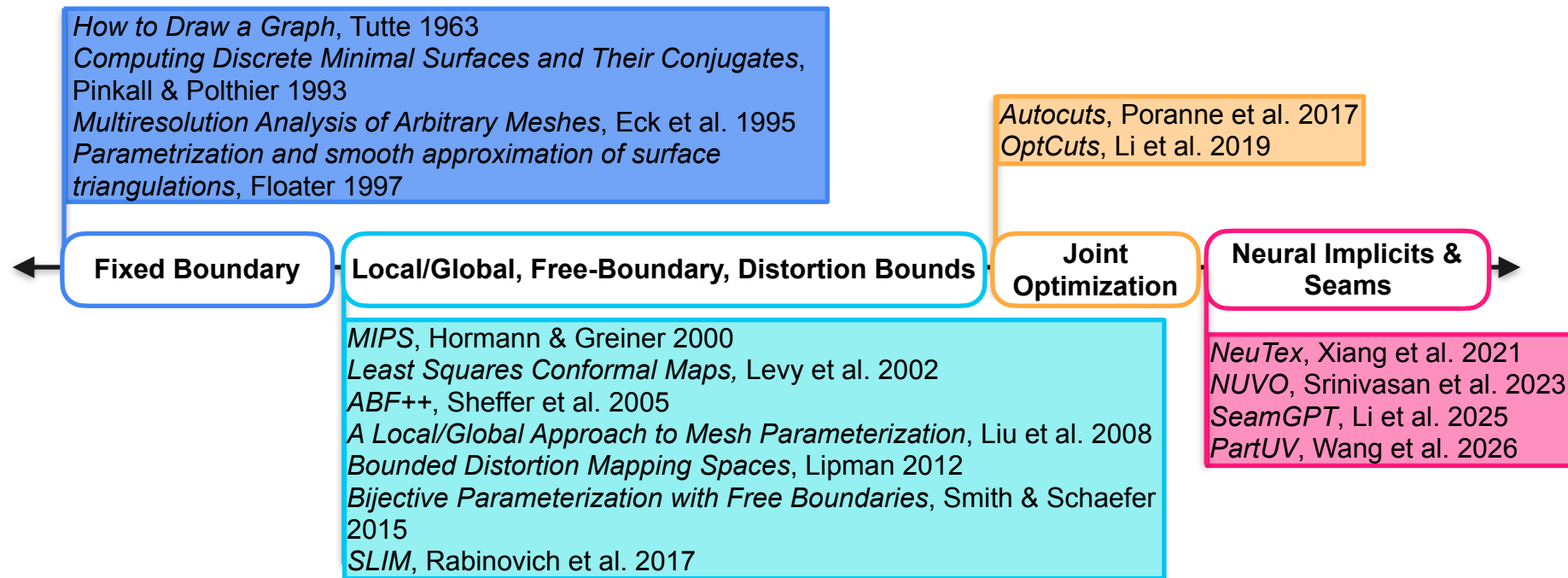
[Source](#)

Most Common Application: Texture Mapping

[Source](#)



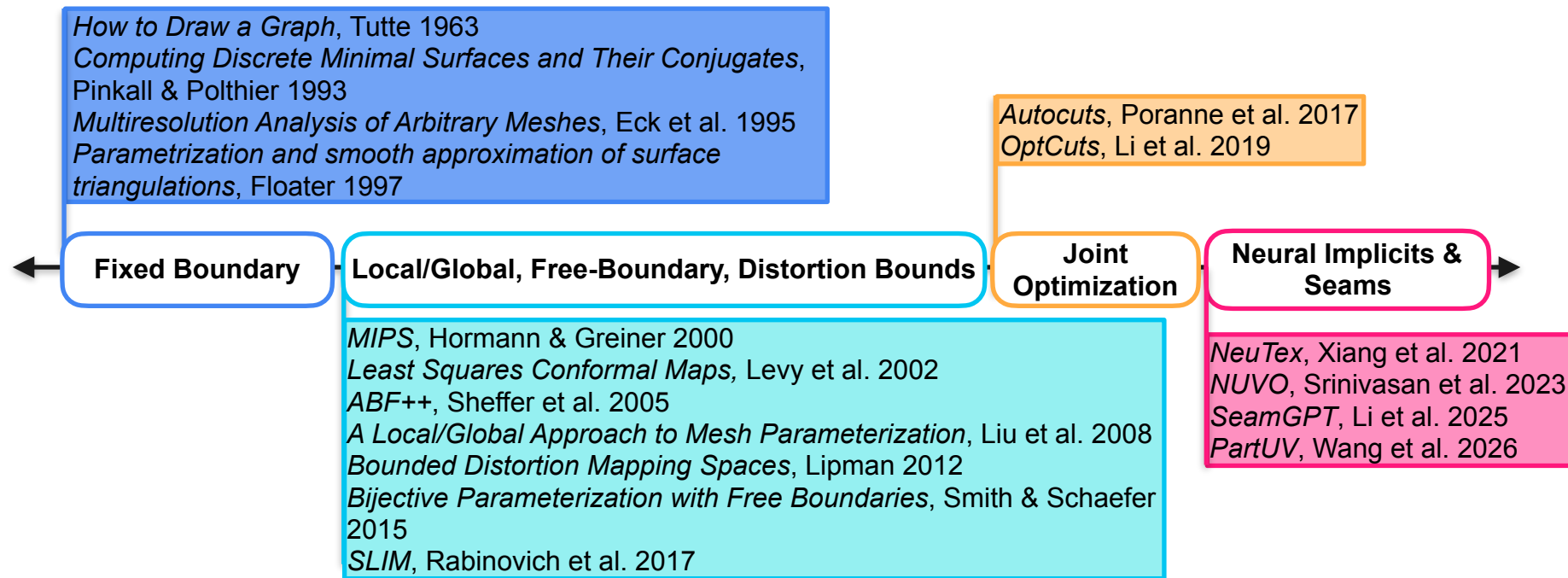
Timeline of Works*



* Not meant to be treated as a comprehensive survey. Important works/subproblems omitted due to presenter's ignorance and commitment to a clean, overlap-free narrative.

Timeline of Works*

You are
here



* Not meant to be treated as a comprehensive survey. Important works/subproblems omitted due to presenter's ignorance and commitment to a clean, overlap-free narrative.

Mathematical Foundations

A **mesh** is a discretization of a smooth surface into a set of vertices, edges, and faces.

Mathematical Foundations

A **mesh** is a discretization of a smooth surface into a set of vertices, edges, and faces.

$$M = (\mathbf{V}, \mathbf{E}, \mathbf{F})$$

Mathematical Foundations

A **mesh** is a discretization of a smooth surface into a set of vertices, edges, and faces.

$$M = (\mathbf{V}, \mathbf{E}, \mathbf{F})$$
$$\mathbf{V} \subset \mathbb{R}^3$$

$$\mathbf{E} = \{(i, j) \mid \text{vertex } i \text{ is connected to vertex } j\}$$

$$\mathbf{F} = \{(i, j, k) \mid \text{vertices } i, j, k \text{ share a face}\}$$

Mathematical Foundations

A mesh **parameterization** is defined as a function mapping from vertices to Ω

$$F : \mathbf{V} \subset \mathbb{R}^3 \rightarrow \Omega \subset \mathbb{R}^2$$

Mathematical Foundations

A mesh **parameterization** is defined as a function mapping from vertices to Ω

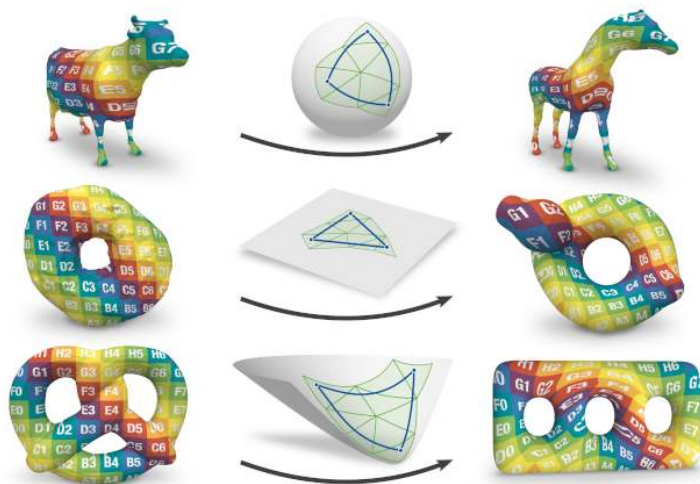
$$F : \mathbf{V} \subset \mathbb{R}^3 \rightarrow \Omega \subset \mathbb{R}^2$$

Classic papers define F^{-1} as the parameterization but I find this direction to be more intuitive.

Mathematical Foundations

A mesh **parameterization** is defined as a function mapping from vertices to Ω

$$F : \mathbf{V} \subset \mathbb{R}^3 \rightarrow \Omega \subset \mathbb{R}^2$$



Ω often in 2D but can really be anything!

Mathematical Foundations

A mesh **parameterization** is defined as a function mapping from vertices to Ω

$$F : \mathbf{V} \subset \mathbb{R}^3 \rightarrow \Omega \subset \mathbb{R}^2$$

In the discrete case, we assign a 2D coordinate to every vertex

$$F(\mathbf{x}) = (u_i, v_i)$$

Mathematical Foundations

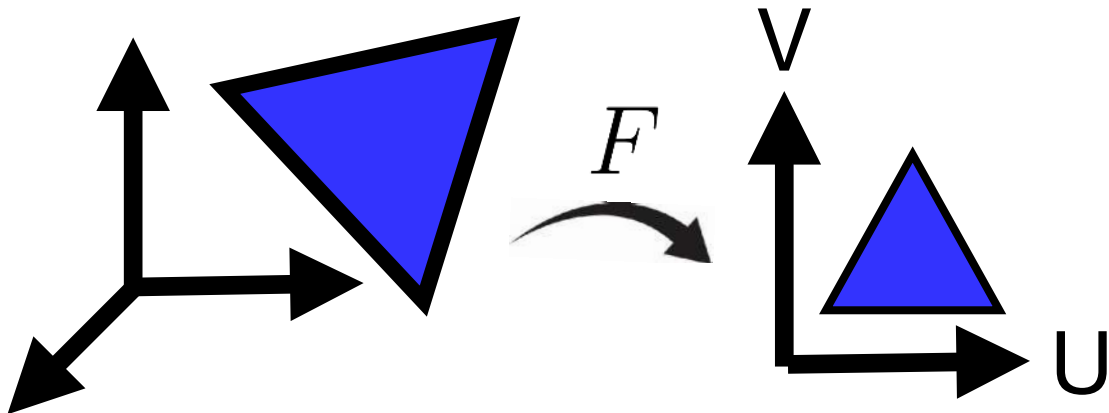
In the discrete case, we assign a 2D coordinate to every vertex

$$F(\mathbf{x}) = (u_i, v_i)$$

Mathematical Foundations

In the discrete case, we assign a 2D coordinate to every vertex

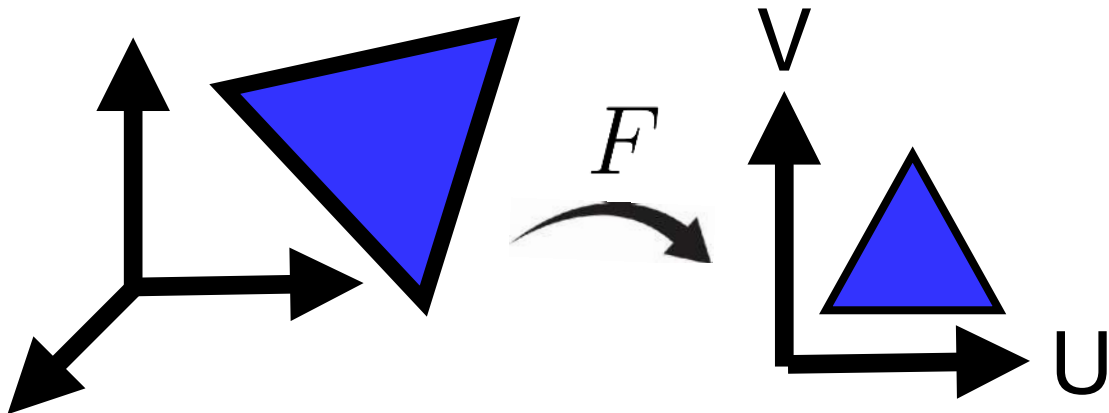
$$F(\mathbf{x}) = (u_i, v_i)$$



Mathematical Foundations

In the discrete case, we assign a 2D coordinate to every vertex

$$F(\mathbf{x}) = (u_i, v_i)$$



2D coordinates
commonly referred to by
“U” and “V” axes,
hence the
colloquialism **UV**
coordinates or **UVs**

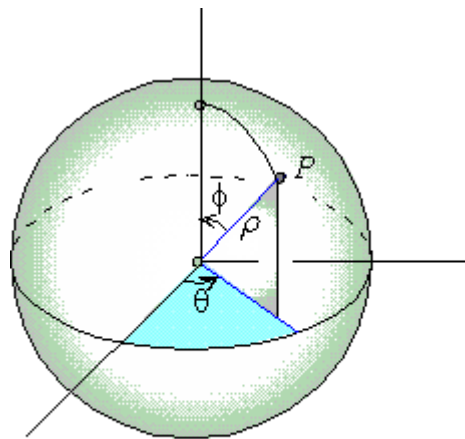
Common Parameterization: Spherical Coordinates

For a sphere with radius R , we have the parameterization

$$U(x, y, z) = \left(\arctan \left(\frac{y}{x} \right), \arccos \left(\frac{z}{R} \right) \right) = (\theta, \phi)$$

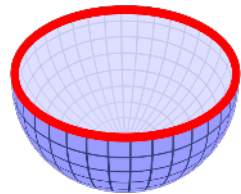
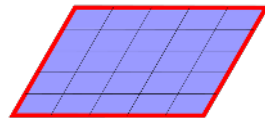
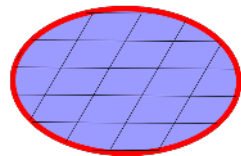
$$0 \leq \theta < 2\pi$$

$$0 \leq \phi < \pi$$



Disk Topology Assumption

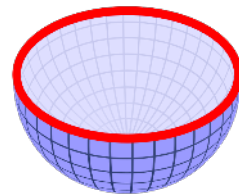
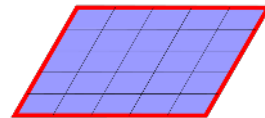
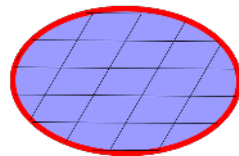
To compute the parameterization, we need to assume our surface is **disk topology**.



Disk Topology Assumption

To compute the parameterization, we need to assume our surface is **disk topology**.

Disk topology: surface with 1 boundary loop.



[Source](#)

Disk Topology Assumption

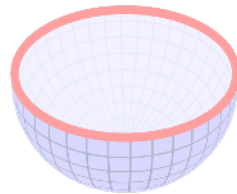
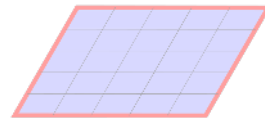
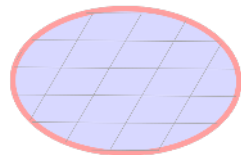
To compute the parameterization, we need to assume our surface is **disk topology**.

Disk topology

Topology 101

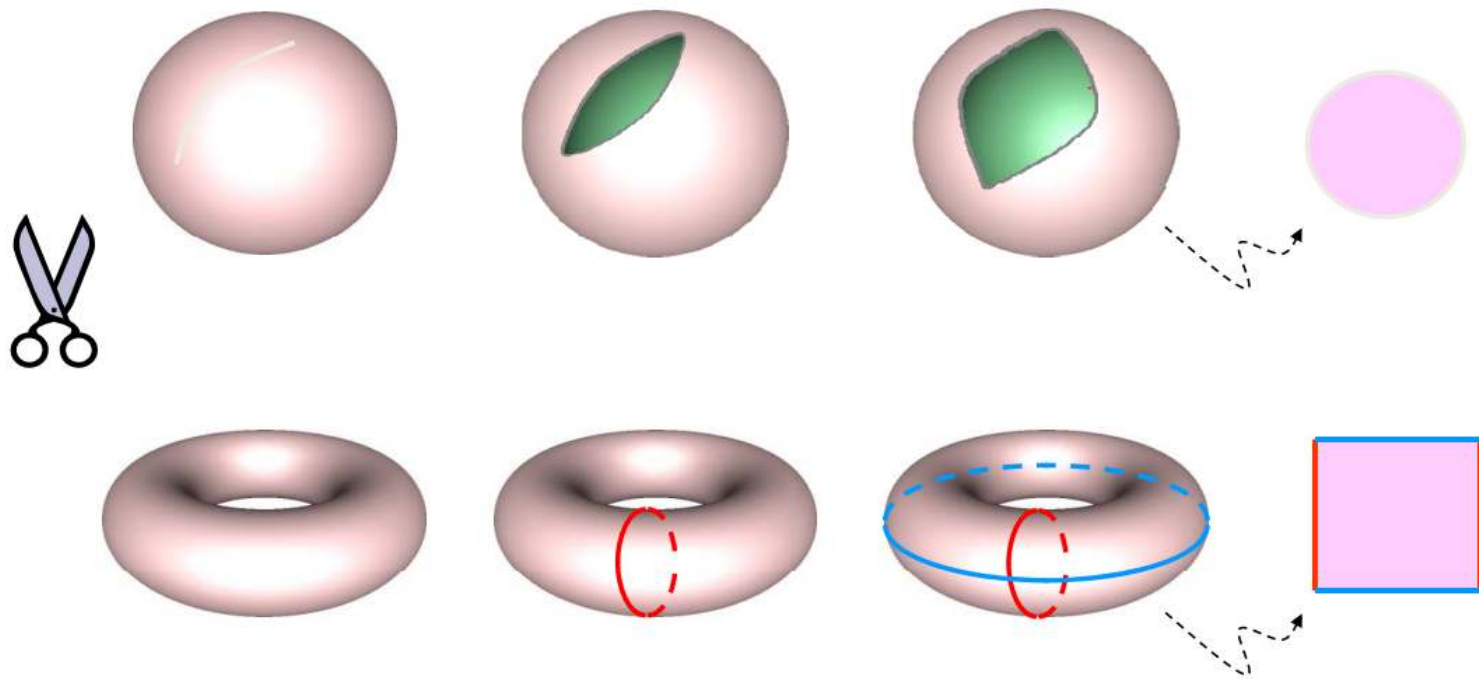
bijjective map (homeomorphism) to the 2D plane.

We can always cut the mesh such that it becomes disk topology!



[Source](#)

Disk Topology Assumption

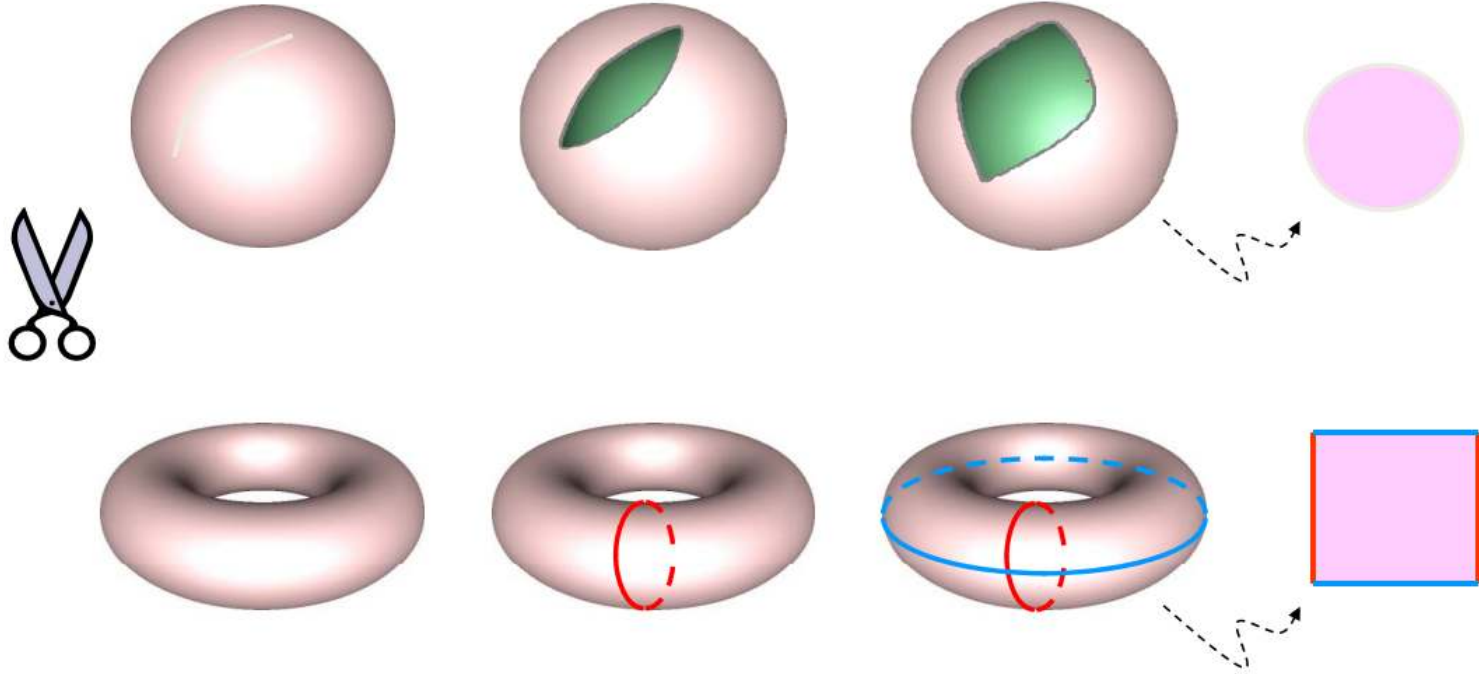


Creates artificial boundary

[Source](#)

Disk Topology Assumption

The problem with seams is they create **discontinuities**.



Creates artificial boundary

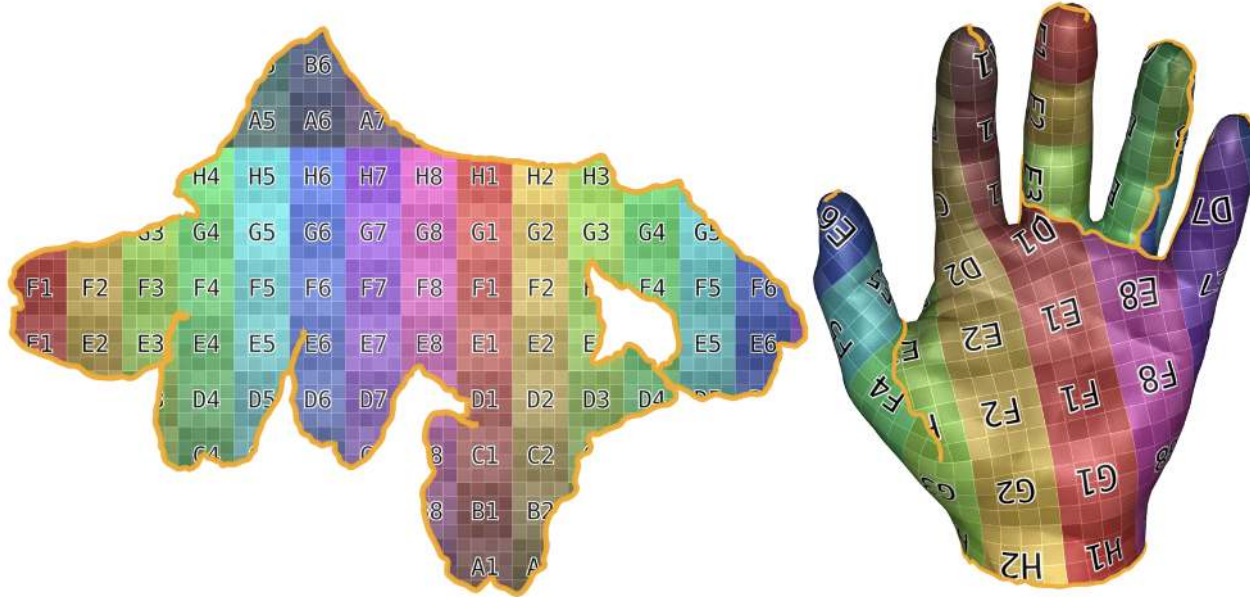
[Source](#)

Disk Topology Assumption

The problem with seams is they create **discontinuities**.

Disk Topology Assumption

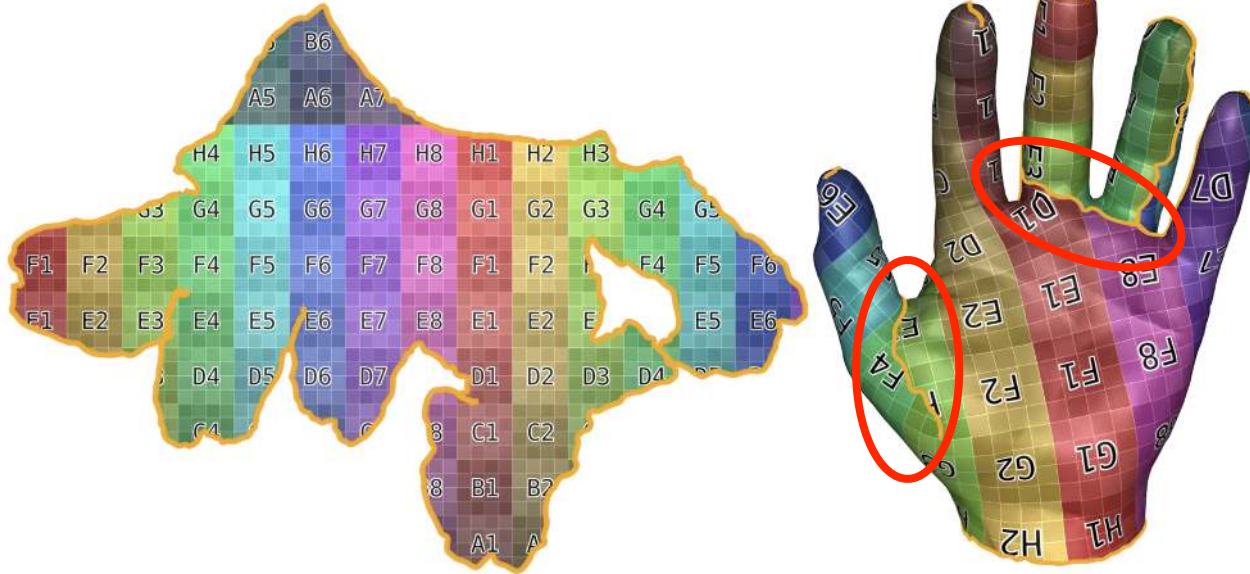
The problem with seams is they create **discontinuities**.



[Source](#)

Disk Topology Assumption

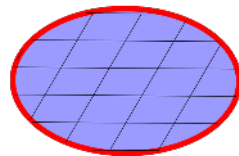
The problem with seams is they create **discontinuities**.



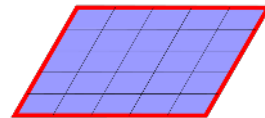
[Source](#)

Disk Topology Assumption

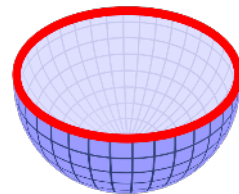
To compute the parameterization, we need to assume our surface is **disk topology**.



Disk topology: surface with 1 boundary loop.



Topology 101. Disk topology allows for **continuous** and **bijective** map (homeomorphism) to the 2D plane.



[Source](#)

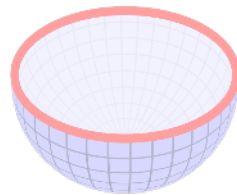
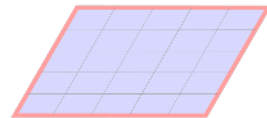
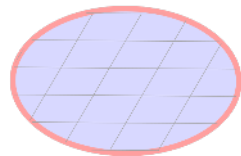
Disk Topology Assumption

To compute the parameterization, we need to assume our surface is **disk topology**.

Disk topology: surface with 1 boundary loop

Not continuous **across** disk boundary

Topology 101. Disk topology allows for **continuous** and **bijective** map (homeomorphism) to the 2D plane.



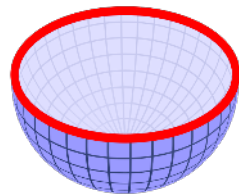
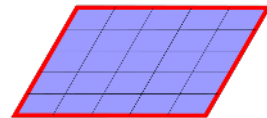
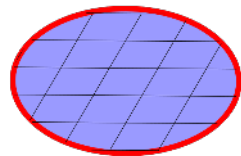
[Source](#)

Disk Topology Assumption

To compute the parameterization, we need to assume our surface is **disk topology**.

Disk topology: surface with 1 boundary loop.

Topology 101. Disk topology allows for **continuous** and **bijective** map (homeomorphism) to the 2D plane.



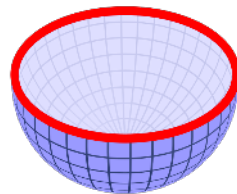
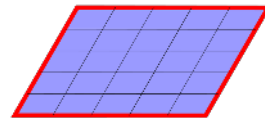
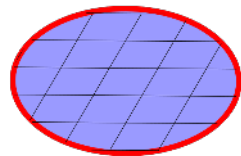
[Source](#)

Disk Topology Assumption

To compute the parameterization, we need to assume our surface is **disk topology**.

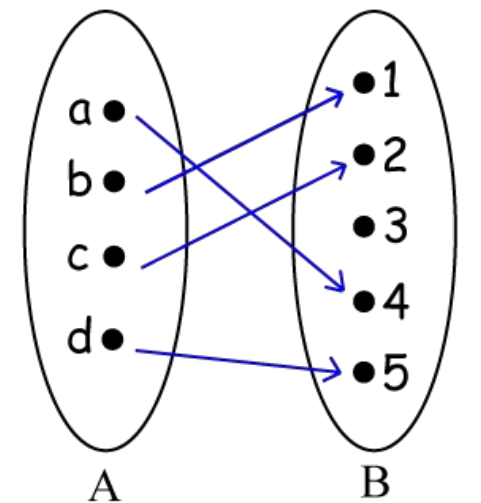
Disk topology: surface with 1 boundary loop.

Topology 101. Disk topology allows for **continuous** and **bijjective** map (homeomorphism) to the 2D plane.



[Source](#)

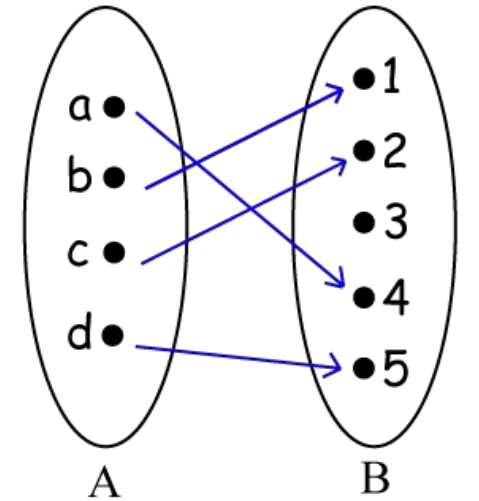
Bijection



Injective
(one-to-one)

[Source](#)

Bijection

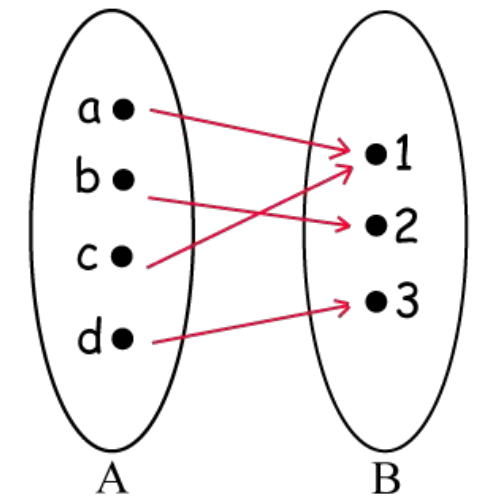


Injective
(one-to-one)

[Source](#)

Every element of A maps to a
unique element of B

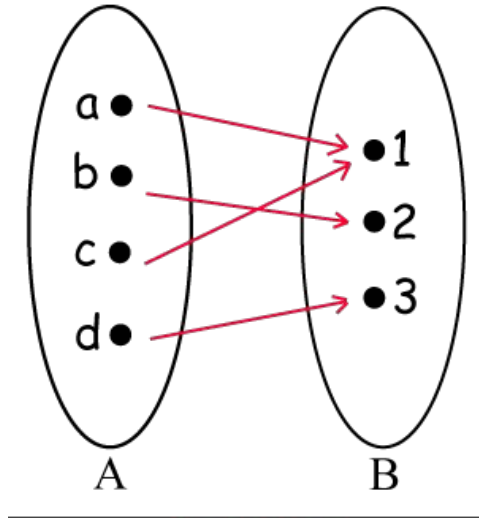
Bijection



Surjective
(onto)

[Source](#)

Bijection

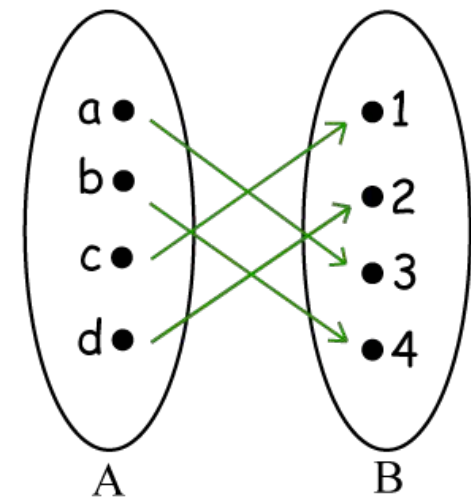


Surjective
(onto)

[Source](#)

Every element of B has **at least one** corresponding element of A

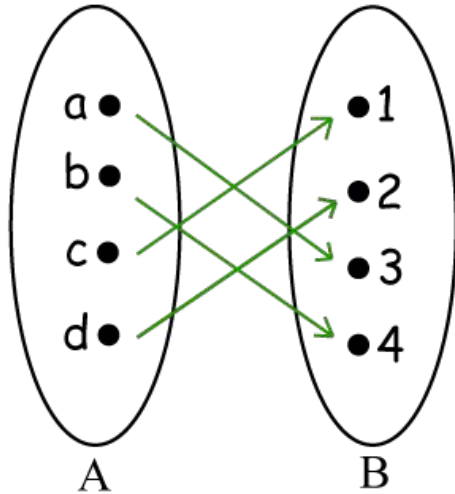
Bijection



Bijjective
(one-to-one and onto)

[Source](#)

Bijection



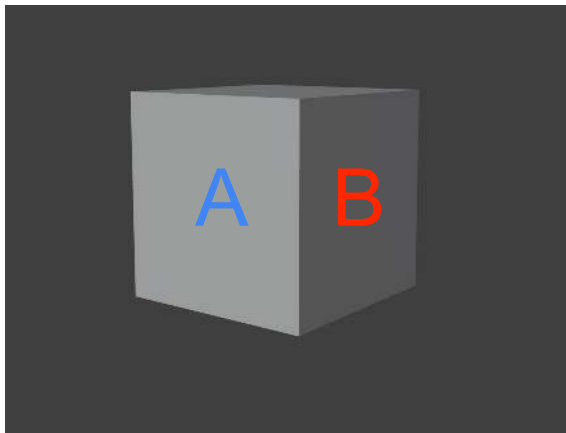
Bijjective
(one-to-one and onto)

[Source](#)

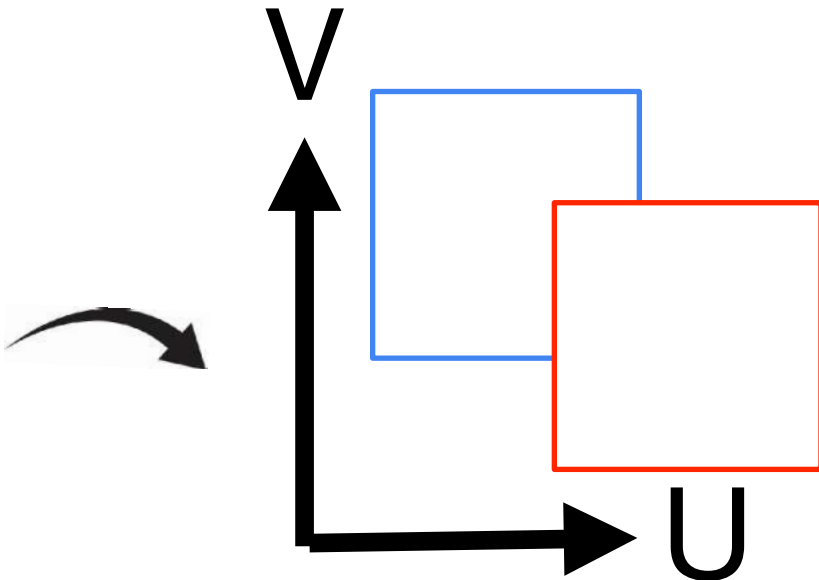
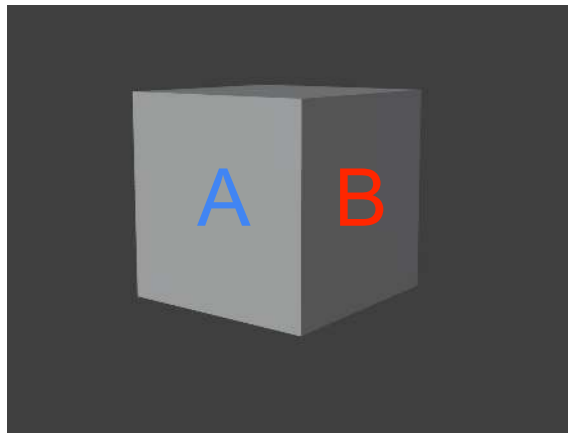
Both injective and surjective.

Exactly matched pairs between
A and B.

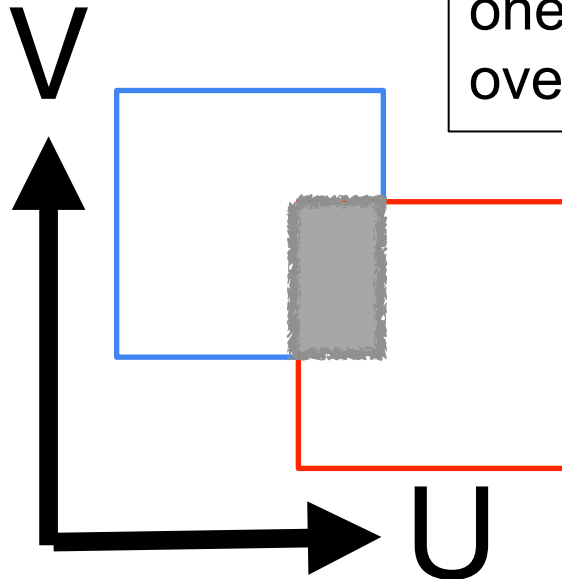
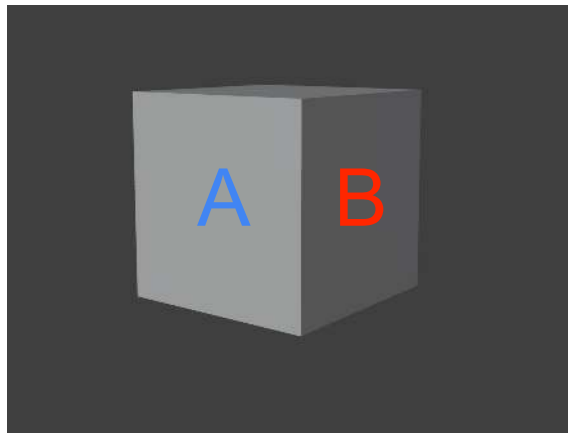
Why Bijectivity (Injectivity) Matters



Why Bijectivity (Injectivity) Matters



Why Bijectivity (Injectivity) Matters

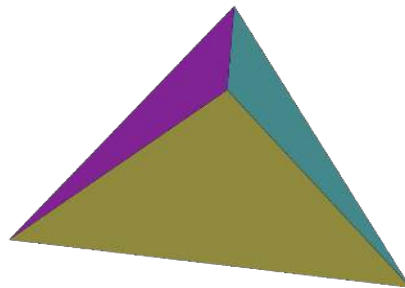


Can only assign
one color to the
overlap.

Part 2: Fixed-Boundary Parameterization

Fixed-Boundary Parameterization

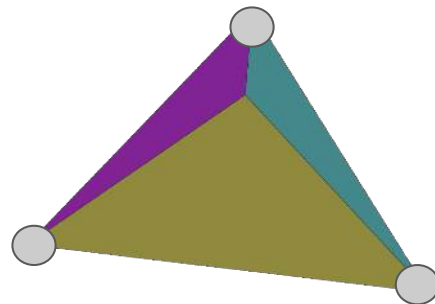
Treat mesh as a **mass-spring system**.



Fixed-Boundary Parameterization

Treat mesh as a **mass-spring system**.

Mesh vertices: nodes with unit mass

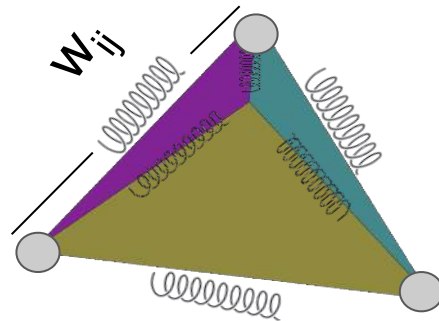


Fixed-Boundary Parameterization

Treat mesh as a **mass-spring system**.

Mesh vertices: nodes with unit mass

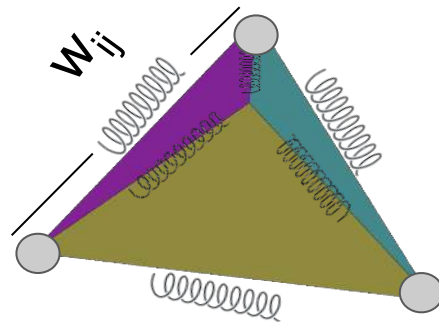
Mesh edges: springs with stiffness constant w_{ij} with zero rest length



Fixed-Boundary Parameterization

Treat mesh as a **mass-spring system**.

Mesh vertices: nodes with unit mass



Mesh edges: springs with stiffness constant w_{ij} with zero rest length

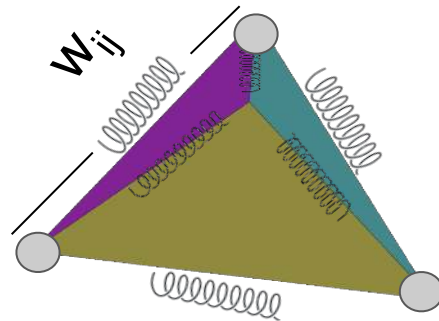
Optimization problem: minimize potential energy of system

$$\min_F \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x}_i) - F(\mathbf{x}_j)||^2$$

Fixed-Boundary Parameterization

Treat mesh as a **mass-spring system**.

Mesh vertices: nodes with unit mass



Mesh edges: springs with stiffness constant w_{ij} with zero rest length

Optimization problem: minimize potential energy of system

$$\min_F \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x}_i) - F(\mathbf{x}_j)||^2$$

“Make every adjacent vertex as close to each other as possible”

Fixed-Boundary Parameterization

Optimization problem: minimize potential energy of system

$$\min_F \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x}_i) - F(\mathbf{x}_j)||^2$$

“Make every adjacent vertex as close to each other as possible”

Fixed-Boundary Parameterization

Optimization problem: minimize potential energy of system

$$\min_F \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x}_i) - F(\mathbf{x}_j)||^2$$

“Make every adjacent vertex as close to each other as possible”

There's a trivial solution to this system. Anyone care to take a guess?

Fixed-Boundary Parameterization

Optimization problem: minimize potential energy of system

$$\min_F \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x}_i) - F(\mathbf{x}_j)||^2$$

“Make every adjacent vertex as close to each other as possible”

There’s a trivial solution to this system. Anyone care to take a guess?

Set $F = \text{constant everywhere}$

Fixed-Boundary Parameterization

Optimization problem: minimize potential energy of system

$$\min_F \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x}_i) - F(\mathbf{x}_j)||^2$$

“Make every adjacent vertex as close to each other as possible”

There’s a trivial solution to this system. Anyone care to take a guess?

Set $F = \text{constant everywhere}$

We need boundary conditions.

Fixed-Boundary Parameterization

Optimization problem: minimize potential energy of system

$$\min_F \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x}_i) - F(\mathbf{x}_j)||^2$$

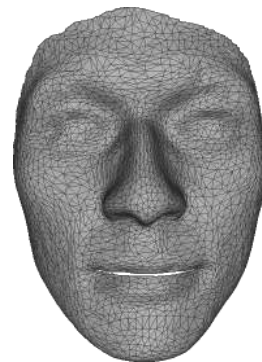
“Make every adjacent vertex as close to each other as possible”

There’s a trivial solution to this system. Anyone care to take a guess?

Set $F = \text{constant everywhere}$

We need boundary conditions.

Pinning the mesh boundary is a natural choice!



Fixed-Boundary Parameterization

Optimization problem: minimize potential energy of system

$$\min_F \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x}_i) - F(\mathbf{x}_j)||^2$$

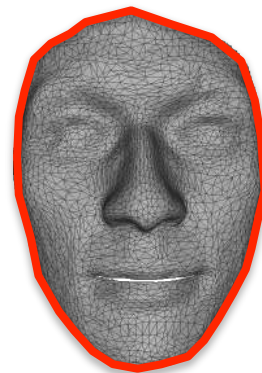
“Make every adjacent vertex as close to each other as possible”

There’s a trivial solution to this system. Anyone care to take a guess?

Set $F = \text{constant everywhere}$

We need boundary conditions.

Pinning the mesh boundary is a natural choice!



Fixed-Boundary Parameterization

Two key design choices to make

$$\min_F \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x}_i) - F(\mathbf{x}_j)||^2$$

Fixed-Boundary Parameterization

Two key design choices to make

$$\min_F \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x}_i) - F(\mathbf{x}_j)||^2$$

1. Where to place mesh boundary

Fixed-Boundary Parameterization

Two key design choices to make

$$\min_F \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x}_i) - F(\mathbf{x}_j)||^2$$

1. Where to place mesh boundary
2. How to define w_{ij}

Fixed-Boundary Parameterization

Two key design choices to make

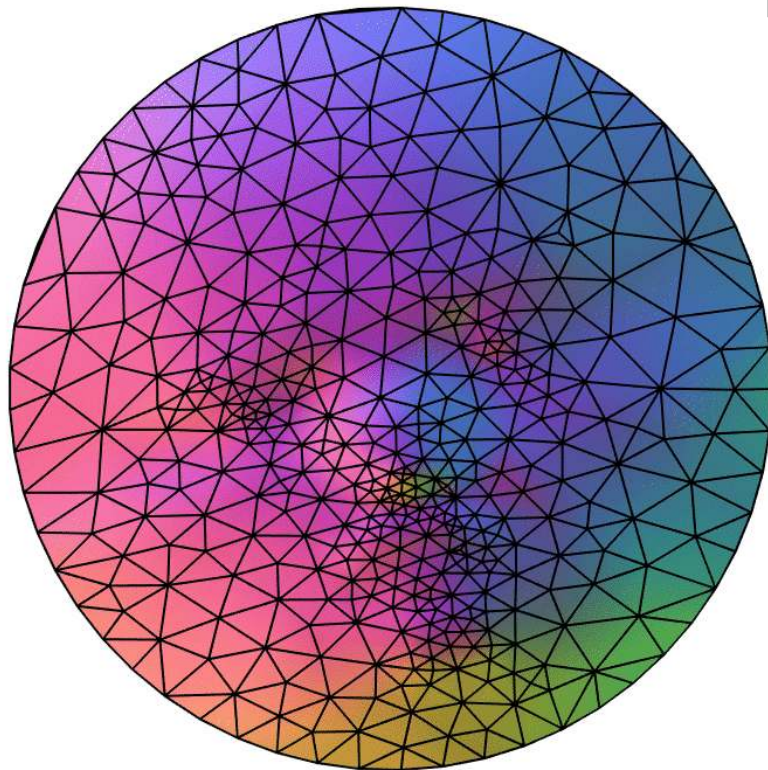
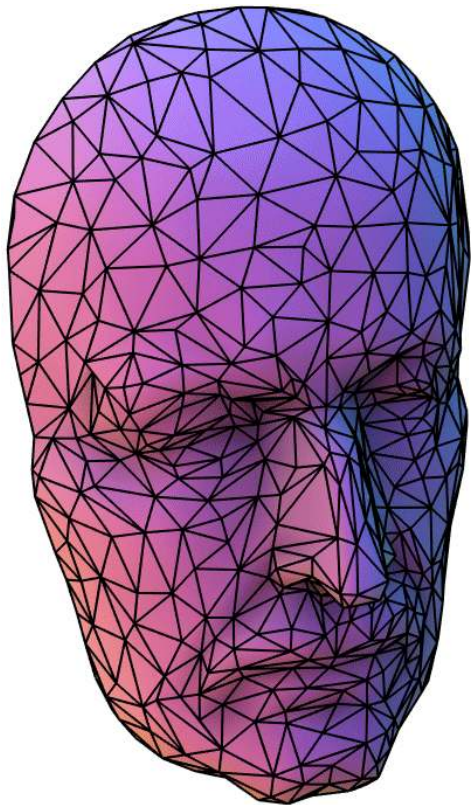
$$\min_F \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x}_i) - F(\mathbf{x}_j)||^2$$

1. Where to place mesh boundary
2. How to define w_{ij}

[Tutte 1963/Floater 1997] If the boundary is **convex** and **weights are positive**, then the solution is guaranteed to be injective.

[Tutte 1963/Floater 1997] If the boundary is **convex** and **weights are positive**, then the solution is guaranteed to be injective.

[Tutte 1963/Floater 1997] If the boundary is **convex** and **weights are positive**, then the solution is guaranteed to be injective.



Tutte embedding

- Boundary is mapped to circle
- All weights are 1

Fixed-Boundary Parameterization

How to solve for F ?

$$E = \sum_{i,j \in \mathbf{E}} w_{ij} ||F_i - F_j||^2$$

Fixed-Boundary Parameterization

How to solve for F?

$$E = \sum_{i,j \in \mathbf{E}} w_{ij} ||F_i - F_j||^2$$

$$\frac{\partial E}{\partial F_i} = \sum_{j \in N_i} 2 * w_{ij} (F_i - F_j) = 0$$

N_i indexes all of the neighbors of vertex i .

Fixed-Boundary Parameterization

How to solve for F ?

$$E = \sum_{i,j \in \mathbf{E}} w_{ij} ||F_i - F_j||^2$$

$$\frac{\partial E}{\partial F_i} = \sum_{j \in N_i} 2 * w_{ij} (F_i - F_j) = 0$$

N_i indexes all of the neighbors of vertex i .

Set boundary vertices and solve for remaining F_i

Fixed-Boundary Parameterization

$$\min_F \sum_{i,j \in \mathbf{E}} w_{ij} ||F_i - F_j||^2$$

Fixed-Boundary Parameterization

$$\min_F \sum_{i,j \in \mathbf{E}} w_{ij} ||F_i - F_j||^2$$

In matrix form, this looks like

$$\min_F \frac{1}{2} \text{tr}(F^T L F)$$

Fixed-Boundary Parameterization

$$\min_F \sum_{i,j \in \mathbf{E}} w_{ij} ||F_i - F_j||^2$$

In matrix form, this looks like

$$\min_F \frac{1}{2} \text{tr}(F^T \boxed{L} F)$$

L is the mesh Laplacian! w_{ij} determines what “type” of discrete Laplacian you’re using. (recall [013_laplacian](#))

Fixed-Boundary Parameterization

$$\min \sum w_{ij} \|F_i - F_j\|^2$$

L is sparse, so the system is sparse.

We will explore the effects of different choice of weights in the exercises.

L is the mesh Laplacian! w_{ij} determines what “type” of discrete Laplacian you’re using. (recall [013_laplacian](#))

Part 3: Computing Parameterization Distortion

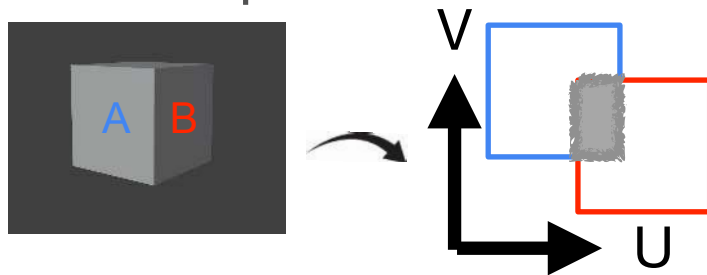
Desiderata Review

Desirable properties of parameterizations

Desiderata Review

Desirable properties of parameterizations

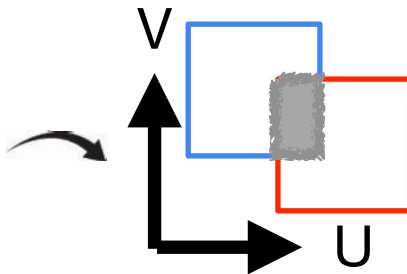
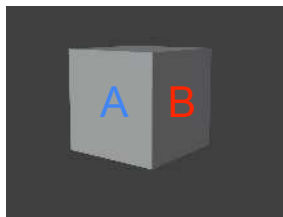
- Injectivity



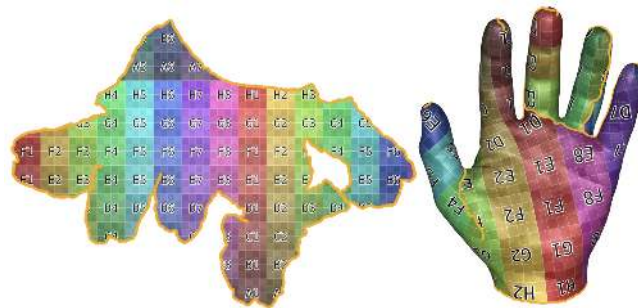
Desiderata Review

Desirable properties of parameterizations

- Injectivity



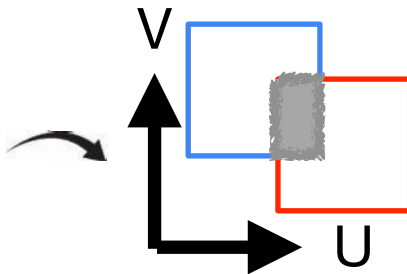
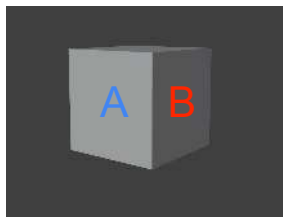
- Seam (discontinuity) minimization



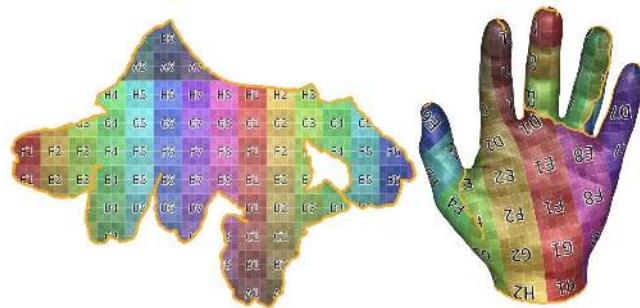
Desiderata Review

Desirable properties of parameterizations

- Injectivity

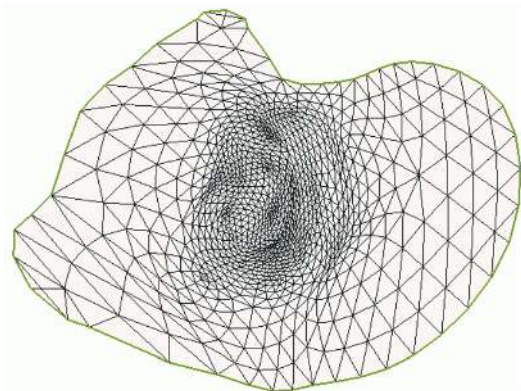


- Seam (discontinuity) minimization

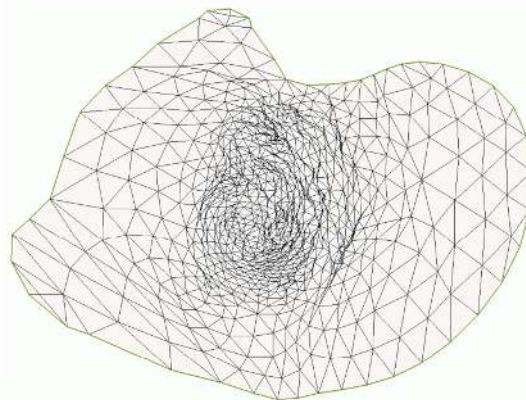


- Low distortion

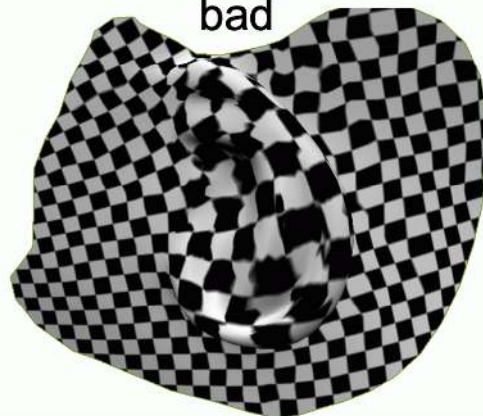
Types of Distortion



good

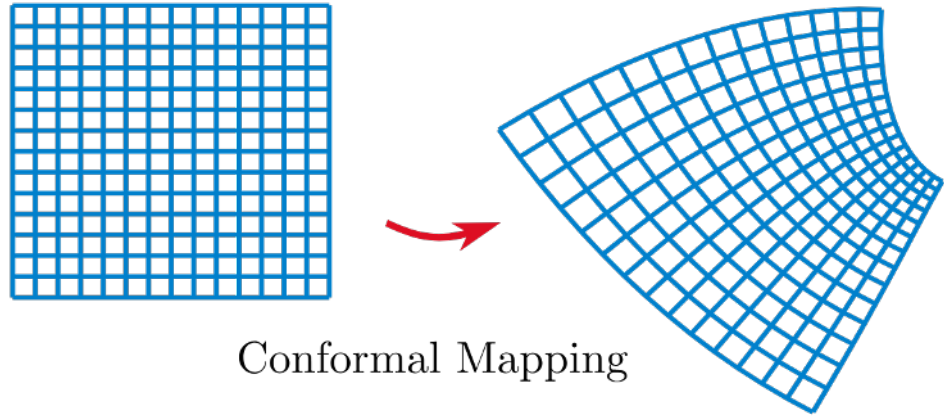


bad



Types of Distortion

Conformal: angle distortion



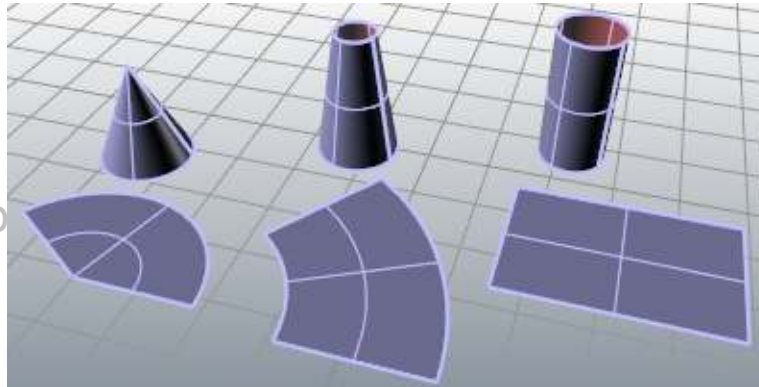
Conformal Mapping

Types of Distortion

Conformal: angle distortion

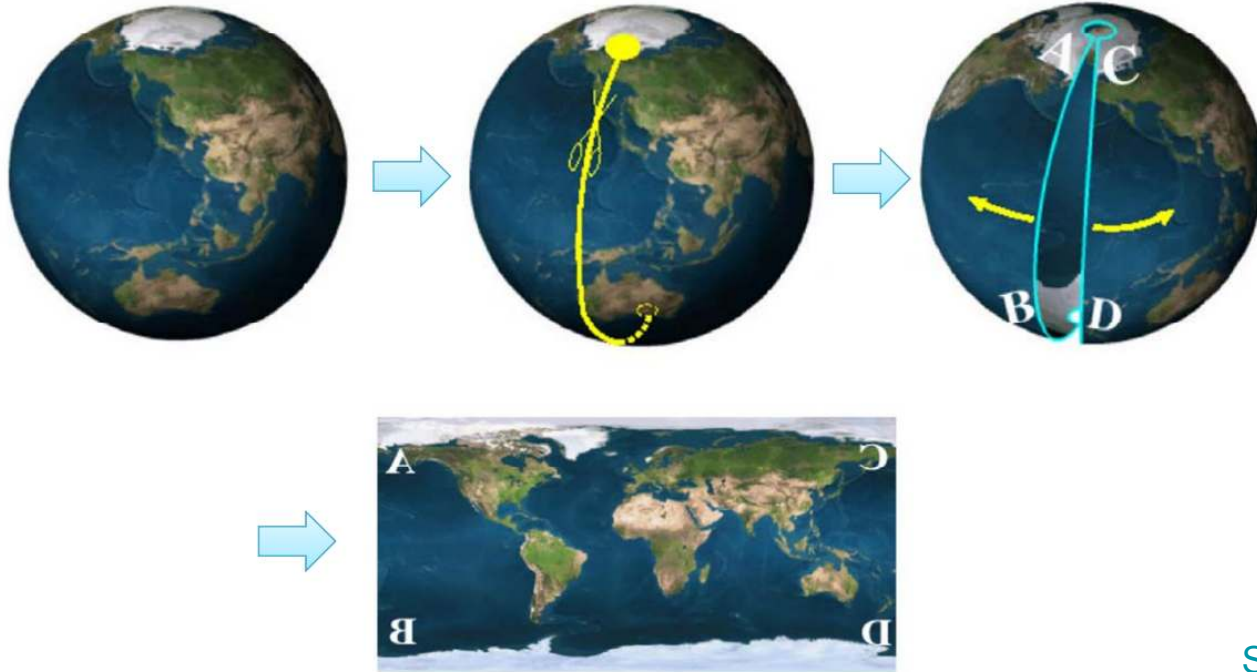
Isometric maps: both angle and area (length) preserving

Authalic: area distortion



Authalic Mapping

Distortion: Motivations in Cartography

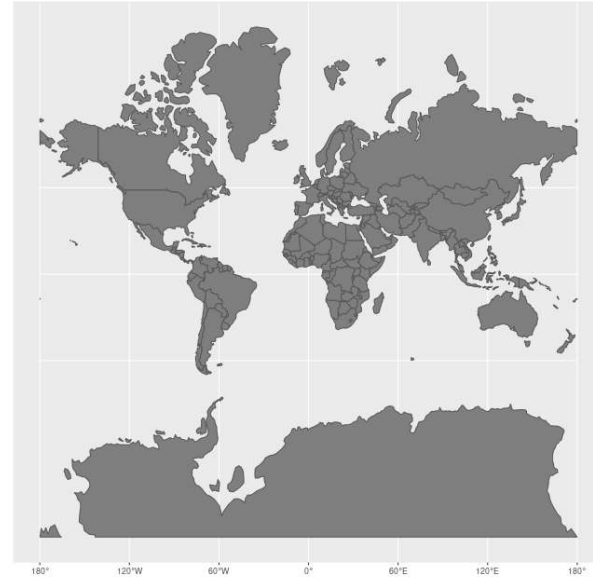


[Source](#)

Distortion: Motivations in Cartography

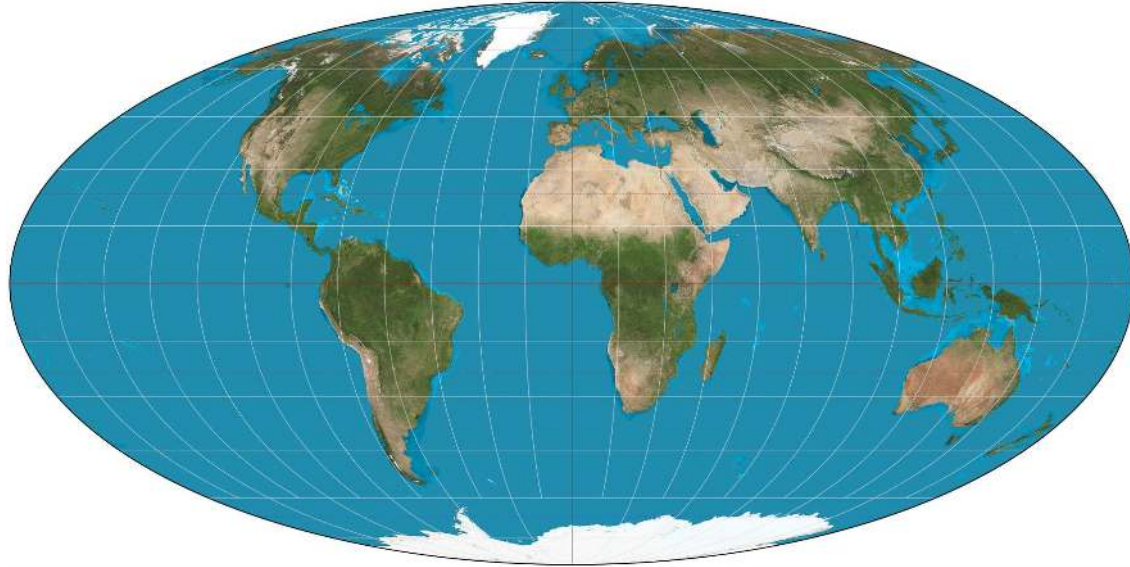


Mercator projection
(conformal)



[Source](#)

Distortion: Motivations in Cartography



Mollweide projection
(equiareal/authalic)

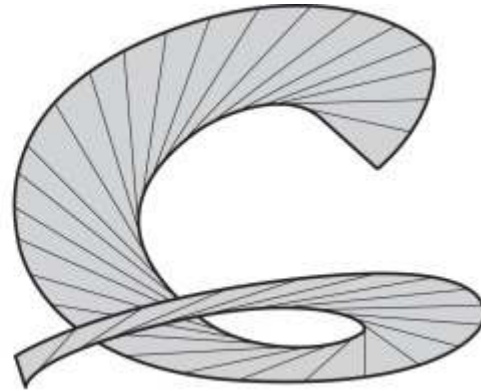
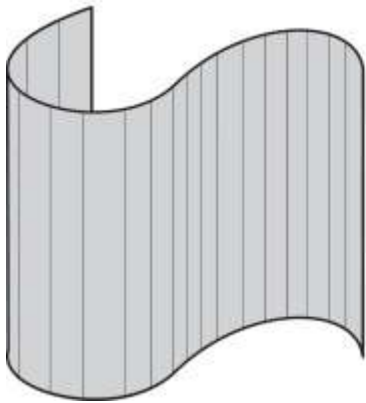
Distortion: Motivations in Cartography

Why are there no isometric (distortionless) maps of the world?

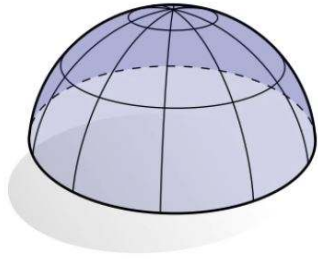
Distortion: Motivations in Cartography

Why are there no isometric (distortionless) maps of the world?

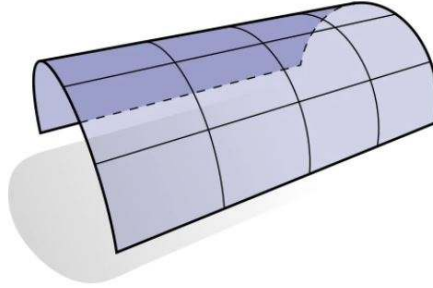
Theorema Egregium (Gauss 1827) Only developable (zero Gaussian curvature) surfaces can be isometrically flattened to the plane.



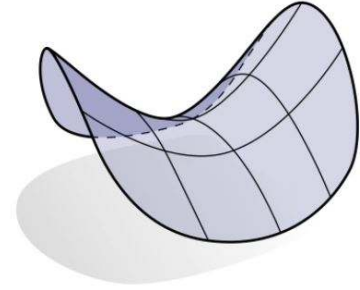
Gaussian Curvature



$$K > 0$$



$$K = 0$$



$$K < 0$$

[Source](#)

Sphere has positive gaussian curvature everywhere

Measuring Distortion

Ok if most parameterizations create distortion how do we measure it?

Measuring Distortion

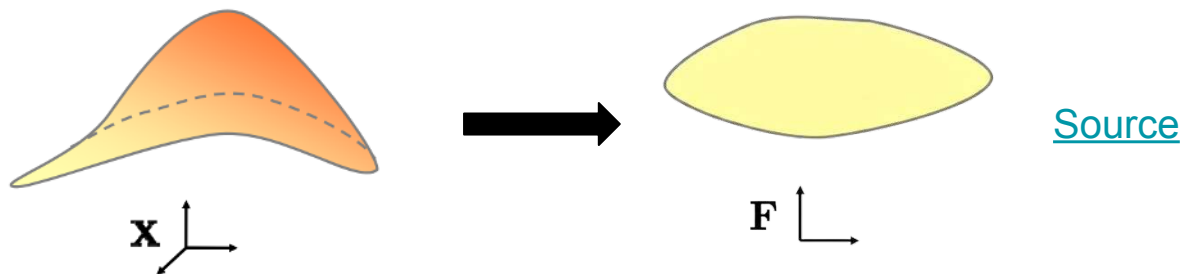
Ok if most parameterizations create distortion how do we measure it?

The Jacobian tells you everything about distortion.

Measuring Distortion

Ok if most parameterizations create distortion how do we measure it?

The Jacobian tells you everything about distortion.



$$F(x, y, z) = \begin{pmatrix} u(x, y, z) \\ v(x, y, z) \end{pmatrix} \quad J_F = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \end{pmatrix}$$

Measuring Distortion

Specifically, the **singular values** of the Jacobian tell you everything about distortion

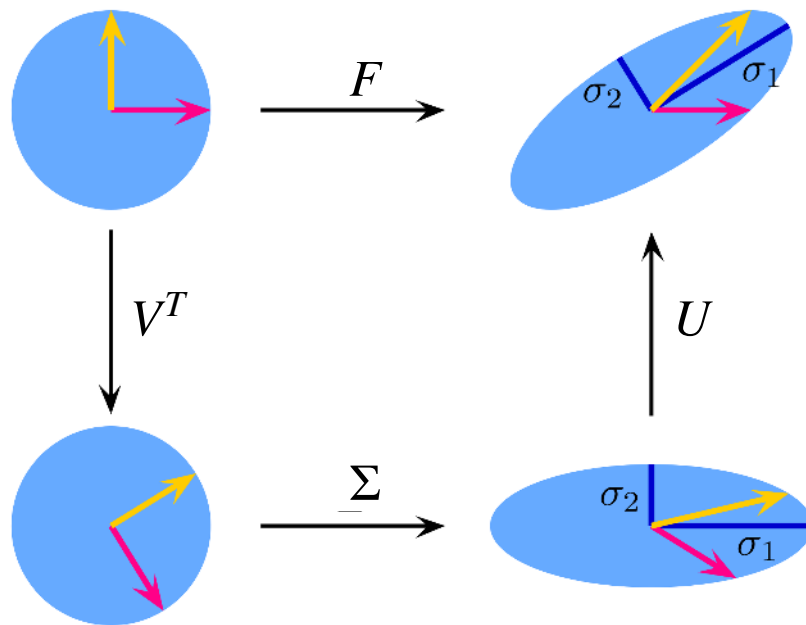
$$J_F = U\Sigma V^T = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} V^T$$

Measuring Distortion

$$J_F = U \Sigma V^T = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} V^T$$

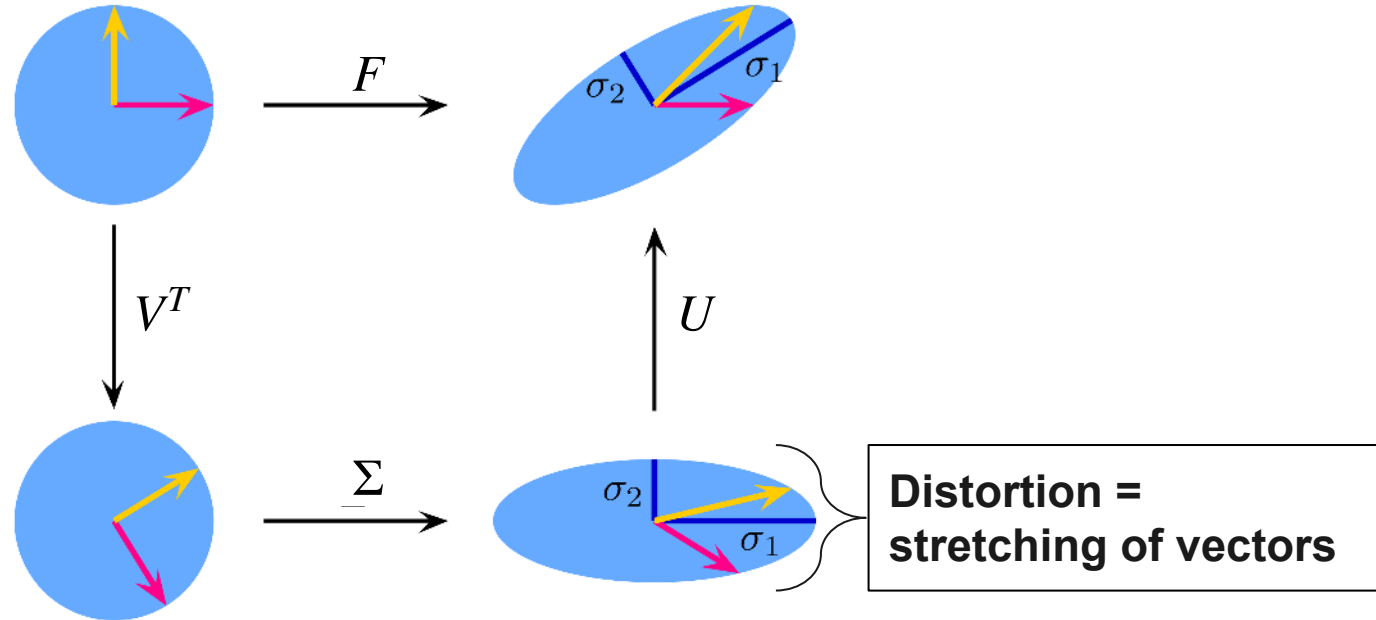
Measuring Distortion

$$J_F = U\Sigma V^T = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} V^T$$



Measuring Distortion

$$J_F = U\Sigma V^T = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} V^T$$



Measuring Distortion

Specifically, the **singular values** of the Jacobian tell you everything about distortion

$$J_F = U\Sigma V^T = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} V^T$$

$$f \text{ is equiareal or area-preserving} \iff \sigma_1 \sigma_2 = 1$$

Measuring Distortion

Specifically, the **singular values** of the Jacobian tell you everything about distortion

$$J_F = U\Sigma V^T = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} V^T$$

$$f \text{ is equiareal or area-preserving} \iff \sigma_1 \sigma_2 = 1$$

$$f \text{ is conformal or angle-preserving} \iff \sigma_1 = \sigma_2$$

Measuring Distortion

Specifically, the **singular values** of the Jacobian tell you everything about distortion

$$J_F = U\Sigma V^T = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} V^T$$

f is *equiareal* or *area-preserving* $\iff \sigma_1\sigma_2 = 1$

f is *conformal* or *angle-preserving* $\iff \sigma_1 = \sigma_2$

f is *isometric* or *length-preserving* $\iff \sigma_1 = \sigma_2 = 1$

Measuring Distortion

Practically, you can find the Jacobian of the UV map function F using the mesh gradient operator $\mathbf{G} \in \mathbb{R}^{3|F| \times |V|}$ (recall [012_gradient](#))

Measuring Distortion

Practically, you can find the Jacobian of the UV map function F using the mesh gradient operator $\mathbf{G} \in \mathbb{R}^{3|F| \times |V|}$ (recall [012_gradient](#))

$$\mathbf{G}F = J_F \in \mathbb{R}^{3|F| \times 2}$$

Measuring Distortion

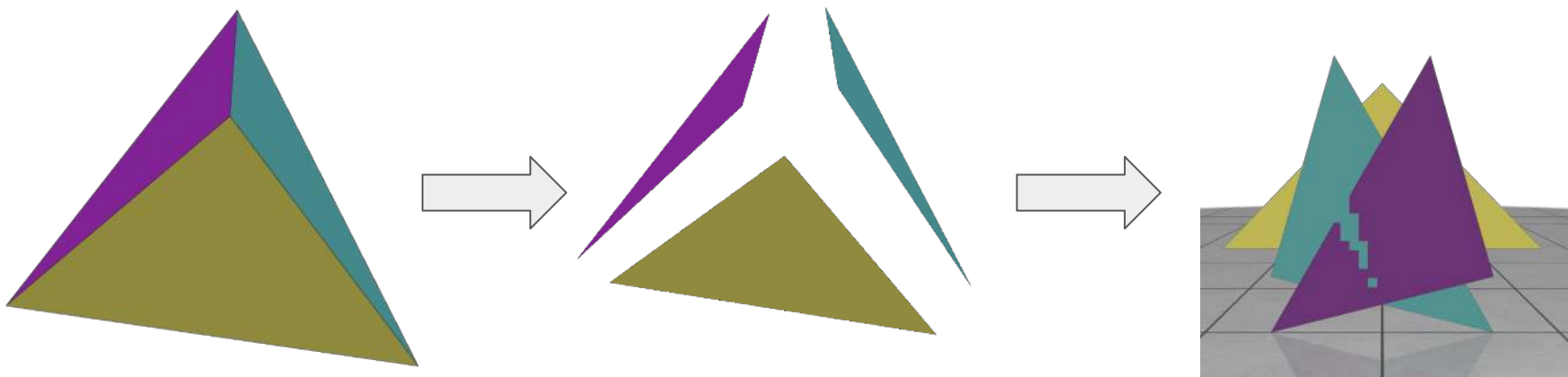
Practically, you can find the Jacobian of the UV map function F using the mesh gradient operator $\mathbf{G} \in \mathbb{R}^{3|F| \times |V|}$ (recall [012_gradient](#))

$$\mathbf{G}F = J_F \in \mathbb{R}^{3|F| \times 2}$$

Jacobian is a 2x3 matrix per triangle

Afternoon Teaser

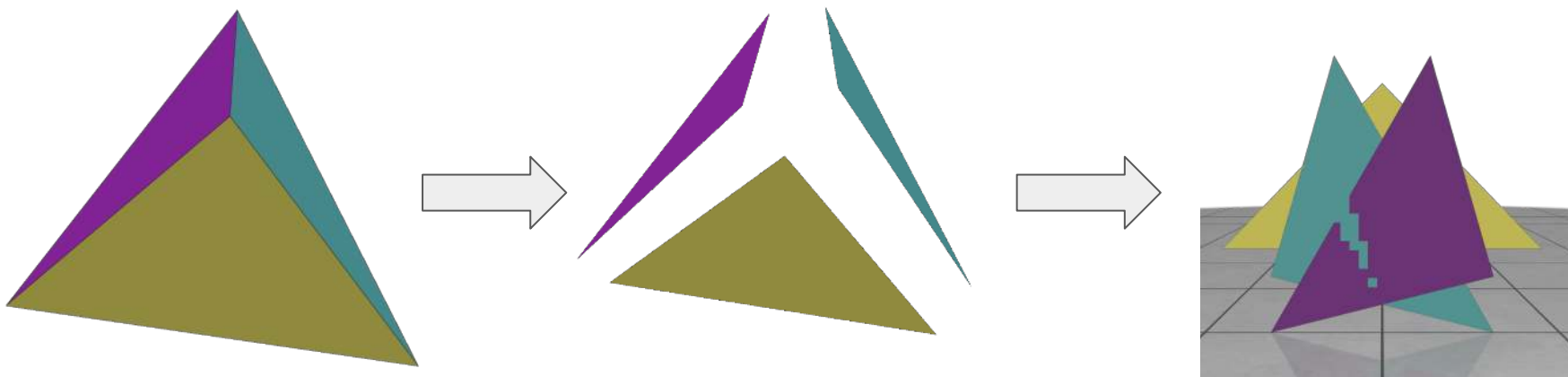
If we split the mesh up into individual triangles (triangle soup) we can trivially obtain a distortionless parameterization.



Afternoon Teaser

If we split the mesh up into individual triangles (triangle soup) we can trivially obtain a distortionless parameterization.

Maximizes discontinuities, so typically a bad solution.

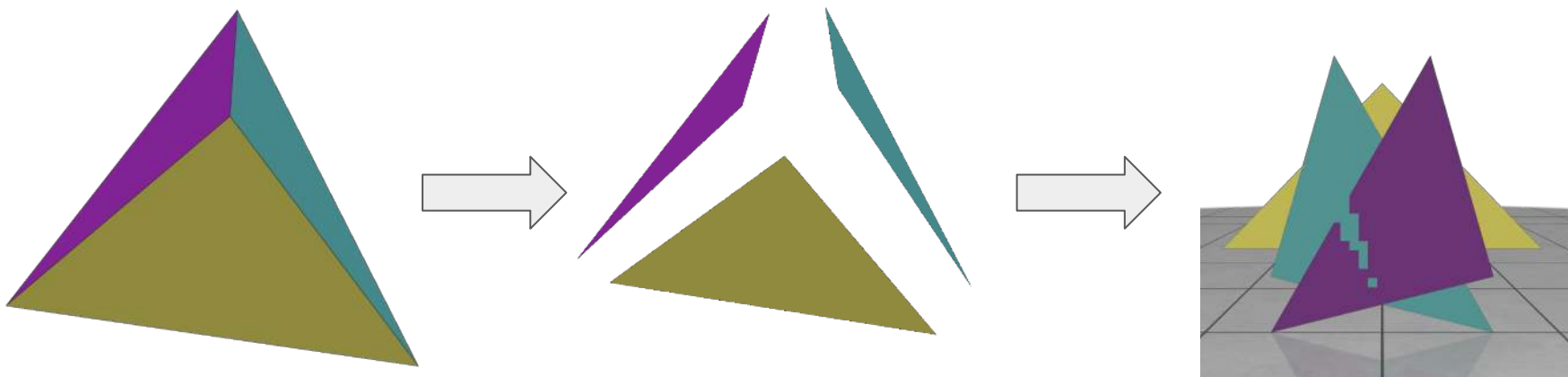


Afternoon Teaser

If we split the mesh up into individual triangles (triangle soup) we can trivially obtain a distortionless parameterization.

Maximizes discontinuities, so typically a bad solution.

Dale will show how we can bypass this discontinuity problem with neural networks (stay tuned!)



Exercises!

Make sure you have the latest version of the repo

```
[(sgi) guanzhi@MacBookPro sgi-introduction-course % git pull  
Already up to date.
```

Exercises!

Make sure you have the latest version of the repo

```
[(sgi) guanzhi@MacBookPro sgi-introduction-course % git pull  
Already up to date.
```

[101_parameterization_fundamentals.ipynb](#): step-by-step walkthrough on how to compute, visualize, and analyze distortion for parameterizations of some simple meshes.

Exercises!

Make sure you have the latest version of the repo

```
[(sgi) guanzhi@MacBookPro sgi-introduction-course % git pull  
Already up to date.
```

[101_parameterization_fundamentals.ipynb](#): step-by-step walkthrough on how to compute, visualize, and analyze distortion for parameterizations of some simple meshes.

[102_parameterization_exercises.ipynb](#): will ask you to apply these same computations on some more complex meshes. And try your hand at some free-boundary methods!

Exercises!

Make sure you have the latest version of the repo

```
[(sgi) guanzhi@MacBookPro sgi-introduction-course % git pull  
Already up to date.
```

[101_parameterization_fundamentals.ipynb](#): step-by-step walkthrough on how to compute, visualize, and analyze distortion for parameterizations of some simple meshes.

[102_parameterization_exercises.ipynb](#): will ask you to apply these same computations on some more complex meshes. And try your hand at some free-boundary methods!

Solution code for reference is in [102_solution.ipynb](#)

Exercises!

Make sure you have the latest version of the repo

```
[(sgi) guanzh  
Already up t
```

[101_parameterization_exercises.ipynb](#)

compute, visu
meshes.

Lots of stuff but most of it is copy-paste work. The goal is exploratory: skip around as you please and try to build some intuition. See if you can parameterize some of your own meshes!

h on how to
ome simple

[102_parameterization_exercises.ipynb](#): will ask you to apply these same computations on some more complex meshes. And try your hand at some free-boundary methods!

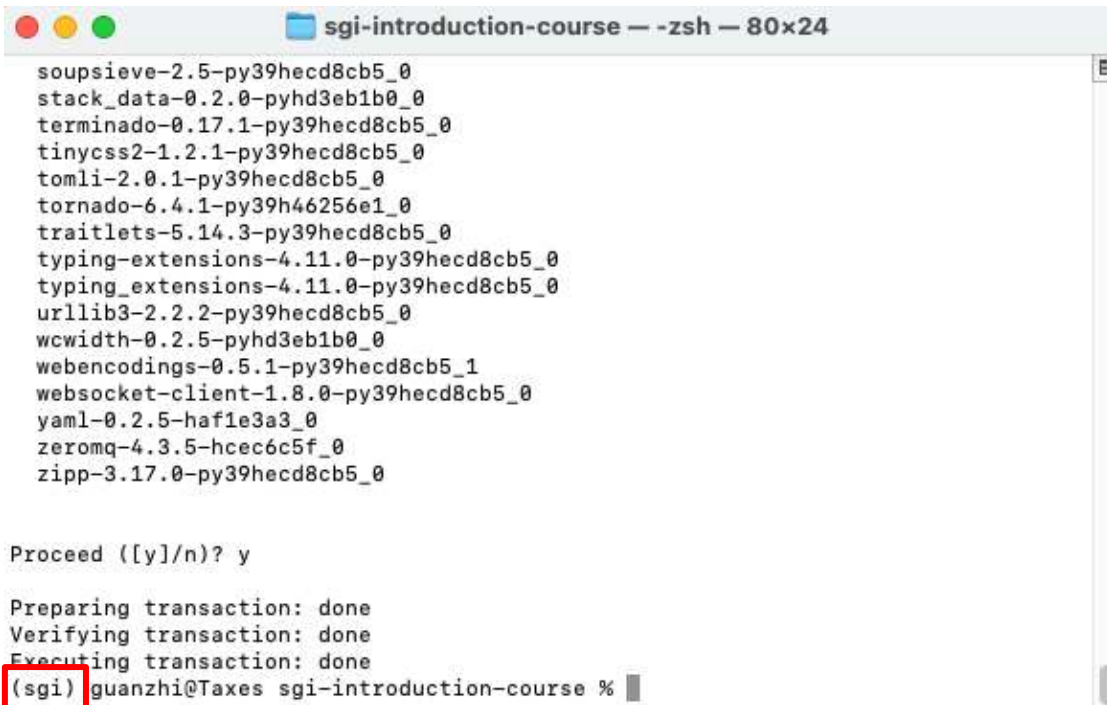
Solution code for reference is in [102_solution.ipynb](#)

Environment Setup

Feel free to skip / check out if you are already comfortable with notebook coding.

Environment Setup

- 1) Make sure your conda/python virtual environment for SGI is activated



A terminal window titled "sgi-introduction-course — -zsh — 80x24" displays the output of a conda environment setup. The output lists various packages and their versions, such as soupsieve-2.5-py39hecd8cb5_0, stack_data-0.2.0-pyhd3eb1b0_0, terminado-0.17.1-py39hecd8cb5_0, tinyccs2-1.2.1-py39hecd8cb5_0, tomli-2.0.1-py39hecd8cb5_0, tornado-6.4.1-py39h46256e1_0, traitlets-5.14.3-py39hecd8cb5_0, typing-extensions-4.11.0-py39hecd8cb5_0, typing_extensions-4.11.0-py39hecd8cb5_0, urllib3-2.2.2-py39hecd8cb5_0, wcwidth-0.2.5-pyhd3eb1b0_0, webencodings-0.5.1-py39hecd8cb5_1, websocket-client-1.8.0-py39hecd8cb5_0, yaml-0.2.5-haf1e3a3_0, zeromq-4.3.5-hcec6c5f_0, and zipp-3.17.0-py39hecd8cb5_0. Below the list, the prompt "Proceed ([y]/n)? y" is shown, followed by "Preparing transaction: done", "Verifying transaction: done", and "Executing transaction: done". The final prompt is "(sgi) guanzhi@Taxes sgi-introduction-course %". A large black arrow points to the "(sgi)" prompt, which is highlighted with a red box.

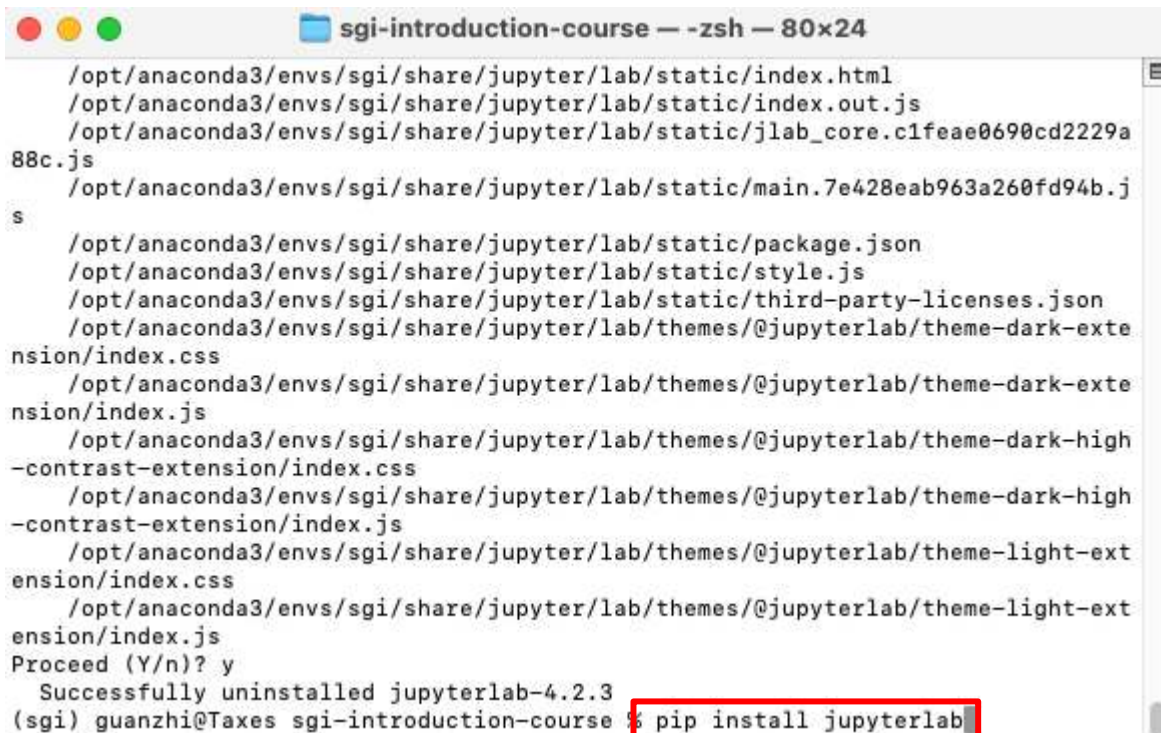
```
sgisieve-2.5-py39hecd8cb5_0
stack_data-0.2.0-pyhd3eb1b0_0
terminado-0.17.1-py39hecd8cb5_0
tinyccs2-1.2.1-py39hecd8cb5_0
tomli-2.0.1-py39hecd8cb5_0
tornado-6.4.1-py39h46256e1_0
traitlets-5.14.3-py39hecd8cb5_0
typing-extensions-4.11.0-py39hecd8cb5_0
typing_extensions-4.11.0-py39hecd8cb5_0
urllib3-2.2.2-py39hecd8cb5_0
wcwidth-0.2.5-pyhd3eb1b0_0
webencodings-0.5.1-py39hecd8cb5_1
websocket-client-1.8.0-py39hecd8cb5_0
yaml-0.2.5-haf1e3a3_0
zeromq-4.3.5-hcec6c5f_0
zipp-3.17.0-py39hecd8cb5_0

Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(sgi) guanzhi@Taxes sgi-introduction-course %
```

Environment Setup

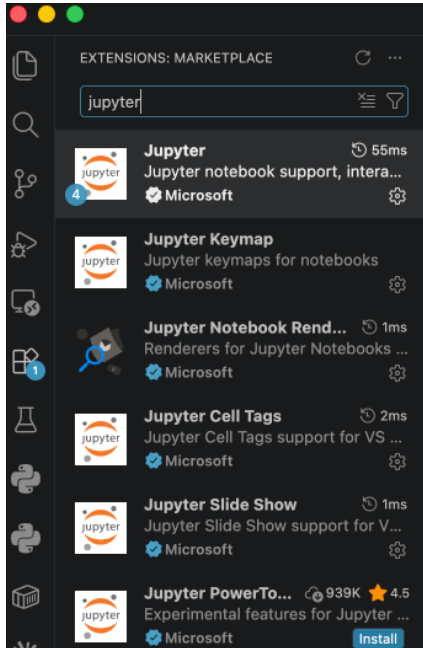
2) pip install jupyterlab



```
sgl-introduction-course — -zsh — 80x24
/opt/anaconda3/envs/sgi/share/jupyter/lab/static/index.html
/opt/anaconda3/envs/sgi/share/jupyter/lab/static/index.out.js
/opt/anaconda3/envs/sgi/share/jupyter/lab/static/jlab_core.c1feae0690cd2229a
88c.js
/opt/anaconda3/envs/sgi/share/jupyter/lab/static/main.7e428eab963a260fd94b.j
s
/opt/anaconda3/envs/sgi/share/jupyter/lab/static/package.json
/opt/anaconda3/envs/sgi/share/jupyter/lab/static/style.js
/opt/anaconda3/envs/sgi/share/jupyter/lab/static/third-party-licenses.json
/opt/anaconda3/envs/sgi/share/jupyter/lab/themes/@jupyterlab/theme-dark-exte
nsion/index.css
/opt/anaconda3/envs/sgi/share/jupyter/lab/themes/@jupyterlab/theme-dark-exte
nsion/index.js
/opt/anaconda3/envs/sgi/share/jupyter/lab/themes/@jupyterlab/theme-dark-high
-contrast-extension/index.css
/opt/anaconda3/envs/sgi/share/jupyter/lab/themes/@jupyterlab/theme-dark-high
-contrast-extension/index.js
/opt/anaconda3/envs/sgi/share/jupyter/lab/themes/@jupyterlab/theme-light-ext
ension/index.css
/opt/anaconda3/envs/sgi/share/jupyter/lab/themes/@jupyterlab/theme-light-ext
ension/index.js
Proceed (Y/n)? y
Successfully uninstalled jupyterlab-4.2.3
(sgi) guanzhi@Taxes sgl-introduction-course % pip install jupyterlab
```

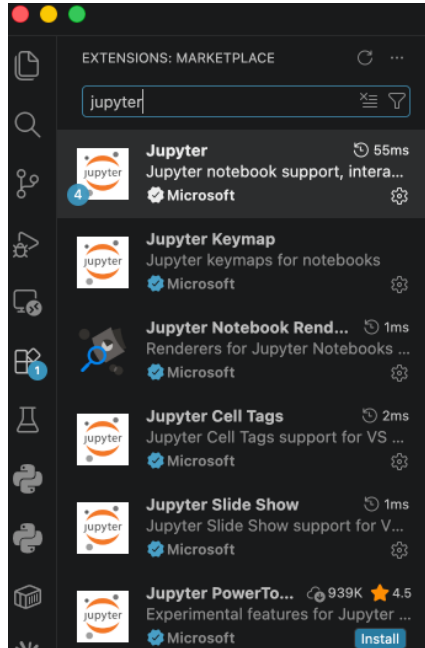
Environment Setup

2.5) If you are on VSCode / some other IDE which has native support for notebook, then can install the Jupyter extension and open the file directly in IDE.

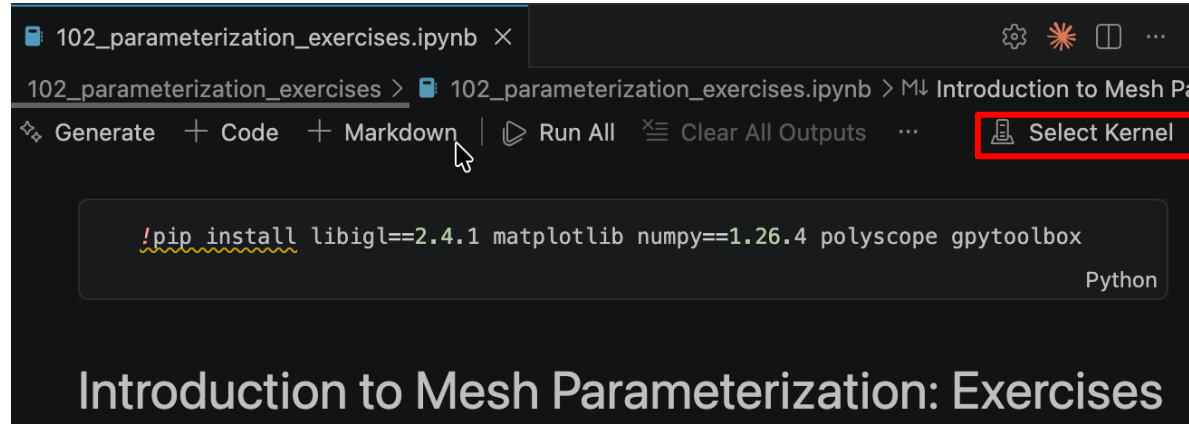


Environment Setup

2.5) If you are on VSCode / some other IDE which has native support for notebook, then can install the Jupyter extension and open the file directly in IDE.



Select the sgi environment when prompted.
May request to install ipykernel if not already installed.



Environment Setup

3) Make sure you are in the sgi-introduction-course directory



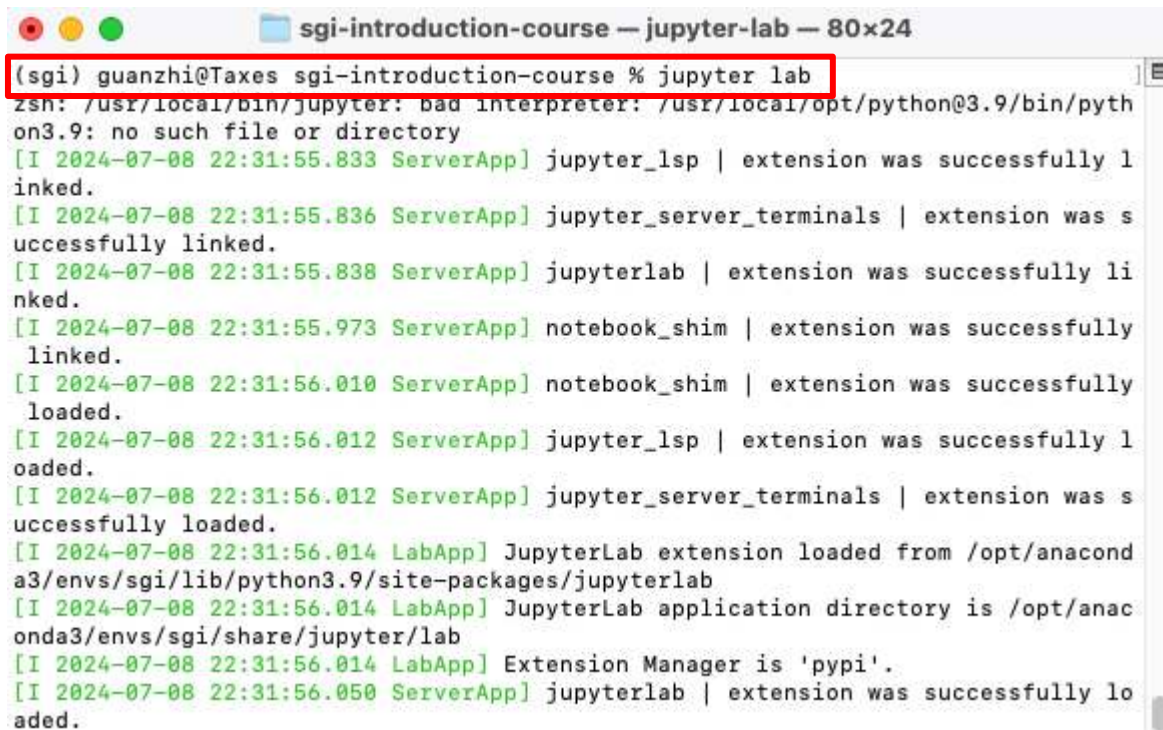
```
sgi-introduction-course — zsh — 80x24

.1.0 argon2-cffi-bindings-21.2.0 arrow-1.3.0 asttokens-2.4.1 async-lru-2.0.4 att
rs-23.2.0 babel-2.15.0 beautifulsoup4-4.12.3 bleach-6.1.0 cffi-1.16.0 charset-no
rmalizer-3.3.2 comm-0.2.2 debugpy-1.8.2 decorator-5.1.1 defusedxml-0.7.1 excepti
ongroup-1.2.1 executing-2.0.1 fastjsonschema-2.20.0 fqdn-1.5.1 h11-0.14.0 httpco
re-1.0.5 httpx-0.27.0 idna-3.7 importlib-metadata-8.0.0 ipykernel-6.29.5 ipython
-8.18.1 isoduration-20.11.0 jedi-0.19.1 jinja2-3.1.4 json5-0.9.25 jsonpointer-3.
0.0 jsonschema-4.23.0 jsonschema-specifications-2023.12.1 jupyter-client-8.6.2 j
upyter-core-5.7.2 jupyter-events-0.10.0 jupyter-lsp-2.2.5 jupyter-server-2.14.1
jupyter-server-terminals-0.5.3 jupyterlab-4.2.3 jupyterlab-pygments-0.3.0 jupyte
rlab-server-2.27.2 matplotlib-inline-0.1.7 mistune-3.0.2 nbclient-0.10.0 nbconve
rt-7.16.4 nbformat-5.10.4 nest-asyncio-1.6.0 notebook-7.2.1 notebook-shim-0.2.4
overrides-7.7.0 packaging-24.1 pandocfilters-1.5.1 parso-0.8.4 pexpect-4.9.0 pla
tformdirs-4.2.2 prometheus-client-0.20.0 prompt-toolkit-3.0.47 psutil-6.0.0 ptyp
rocess-0.7.0 pure-eval-0.2.2 pycparser-2.22 pygments-2.18.0 python-dateutil-2.9.
0.post0 python-json-logger-2.0.7 pyyaml-6.0.1 pyzmq-26.0.3 referencing-0.35.1 re
quests-2.32.3 rfc3339-validator-0.1.4 rfc3986-validator-0.1.1 rpds-py-0.19.0 sen
d2trash-1.8.3 six-1.16.0 sniffio-1.3.1 soupsieve-2.5 stack-data-0.6.3 terminado-
0.18.1 tinycss2-1.3.0 tomli-2.0.1 tornado-6.4.1 traitlets-5.14.3 types-python-da
teutil-2.9.0.20240316 typing-extensions-4.12.2 uri-template-1.3.0 urllib3-2.2.2
wcwidth-0.2.13 webcolors-24.6.0 webencodings-0.5.1 websocket-client-1.8.0 zipp-3
.19.2

(sgi) guanzhi@Taxes sgi-introduction-course % pwd
/Users/guanzhi/Documents/Graphics/sgi-introduction-course
(sgi) guanzhi@Taxes sgi-introduction-course %
```

Environment Setup

4) Call `jupyter lab` to launch the local server



```
sgl-introduction-course — jupyter-lab — 80x24
(sgi) guanzhi@Taxes sgi-introduction-course % jupyter lab
zsh: /usr/local/bin/jupyter: bad interpreter: /usr/local/opt/python@3.9/bin/pyth
on3.9: no such file or directory
[I 2024-07-08 22:31:55.833 ServerApp] jupyter_lsp | extension was successfully l
inked.
[I 2024-07-08 22:31:55.836 ServerApp] jupyter_server_terminals | extension was s
uccessfully linked.
[I 2024-07-08 22:31:55.838 ServerApp] jupyterlab | extension was successfully li
nked.
[I 2024-07-08 22:31:55.973 ServerApp] notebook_shim | extension was successfully
linked.
[I 2024-07-08 22:31:56.010 ServerApp] notebook_shim | extension was successfully
loaded.
[I 2024-07-08 22:31:56.012 ServerApp] jupyter_lsp | extension was successfully l
oaded.
[I 2024-07-08 22:31:56.012 ServerApp] jupyter_server_terminals | extension was s
uccessfully loaded.
[I 2024-07-08 22:31:56.014 LabApp] JupyterLab extension loaded from /opt/anacond
a3/envs/sgi/lib/python3.9/site-packages/jupyterlab
[I 2024-07-08 22:31:56.014 LabApp] JupyterLab application directory is /opt/anac
onda3/envs/sgi/share/jupyter/lab
[I 2024-07-08 22:31:56.014 LabApp] Extension Manager is 'pypi'.
[I 2024-07-08 22:31:56.050 ServerApp] jupyterlab | extension was successfully lo
aded.
```


Environment Setup

4) If the server doesn't start in your browser automatically, then use one of the links provided



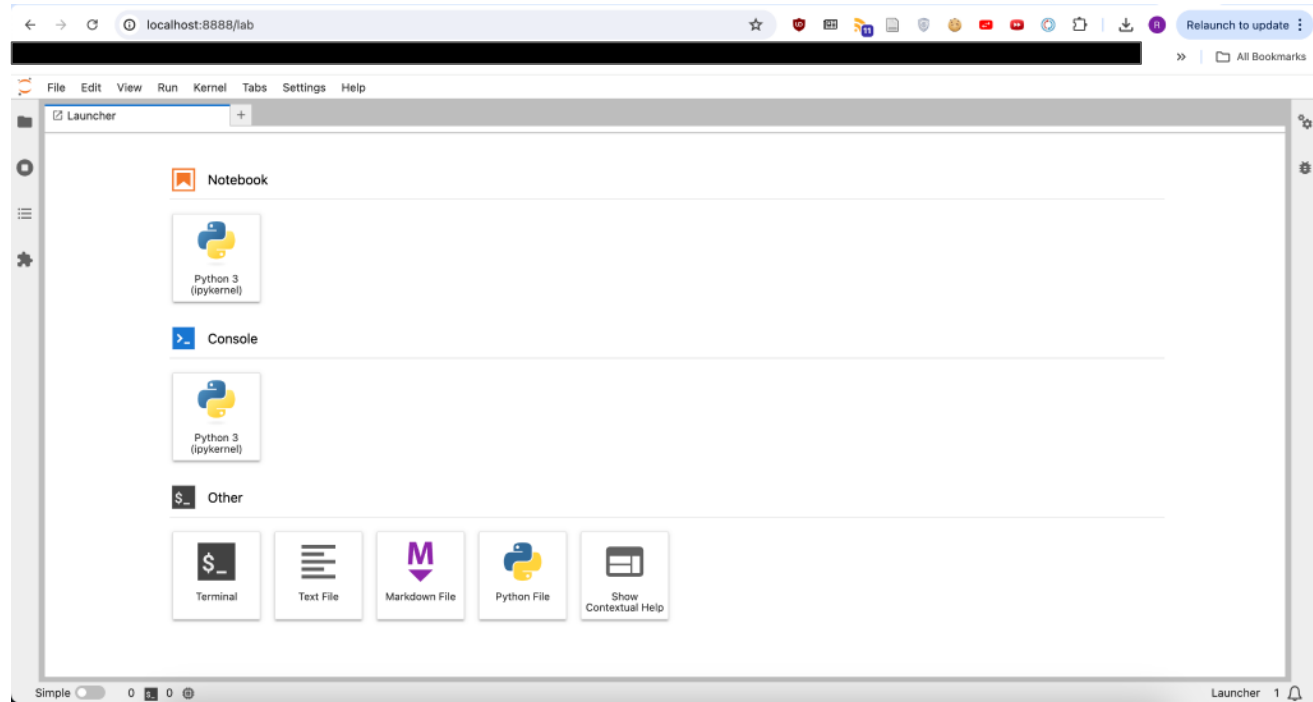
The image shows a terminal window titled "sgi-introduction-course — jupyter-lab — 80x24". The terminal output includes several log messages from the JupyterLab ServerApp. A red rectangular box highlights a section of the output that provides instructions on how to access the server. The instructions state that the user should open a file in a browser or copy and paste one of the provided URLs. The URLs are: `file:///Users/guanzhi/Library/Jupyter/runtime/jpserver-80748-open.html` and `http://localhost:8888/lab?token=3a3fac691999ac0be874b5e6285489fd00ae3797197f9a2a`. Below the highlighted section, there are more log messages, including a warning about the npm prefix and a message about skipping non-installed servers.

```
[I 2024-07-08 22:31:56.051 ServerApp] http://127.0.0.1:8888/lab?token=3a3fac691999ac0be874b5e6285489fd00ae3797197f9a2a
[I 2024-07-08 22:31:56.051 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 2024-07-08 22:31:56.055 ServerApp]

To access the server, open this file in a browser:
file:///Users/guanzhi/Library/Jupyter/runtime/jpserver-80748-open.html
Or copy and paste one of these URLs:
http://localhost:8888/lab?token=3a3fac691999ac0be874b5e6285489fd00ae3797197f9a2a
http://127.0.0.1:8888/lab?token=3a3fac691999ac0be874b5e6285489fd00ae3797197f9a2a
[W 2024-07-08 22:31:56.085 ServerApp] Could not determine npm prefix: Command '['/usr/local/bin/npm', 'prefix', '-g']' returned non-zero exit status 127.
[I 2024-07-08 22:31:56.317 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-nodejs, javascript-typescript-langserver, jedi-language-server, julia-language-server, pyright, python-language-server, python-lsp-server, r-languageserver, sql-language-server, texlab, typescript-language-server, unified-language-server, vscode-css-languageserver-bin, vscode-html-languageserver-bin, vscode-json-languageserver-bin, yaml-language-server
[W 2024-07-08 22:31:58.022 LabApp] Could not determine jupyterlab build status without nodejs
```

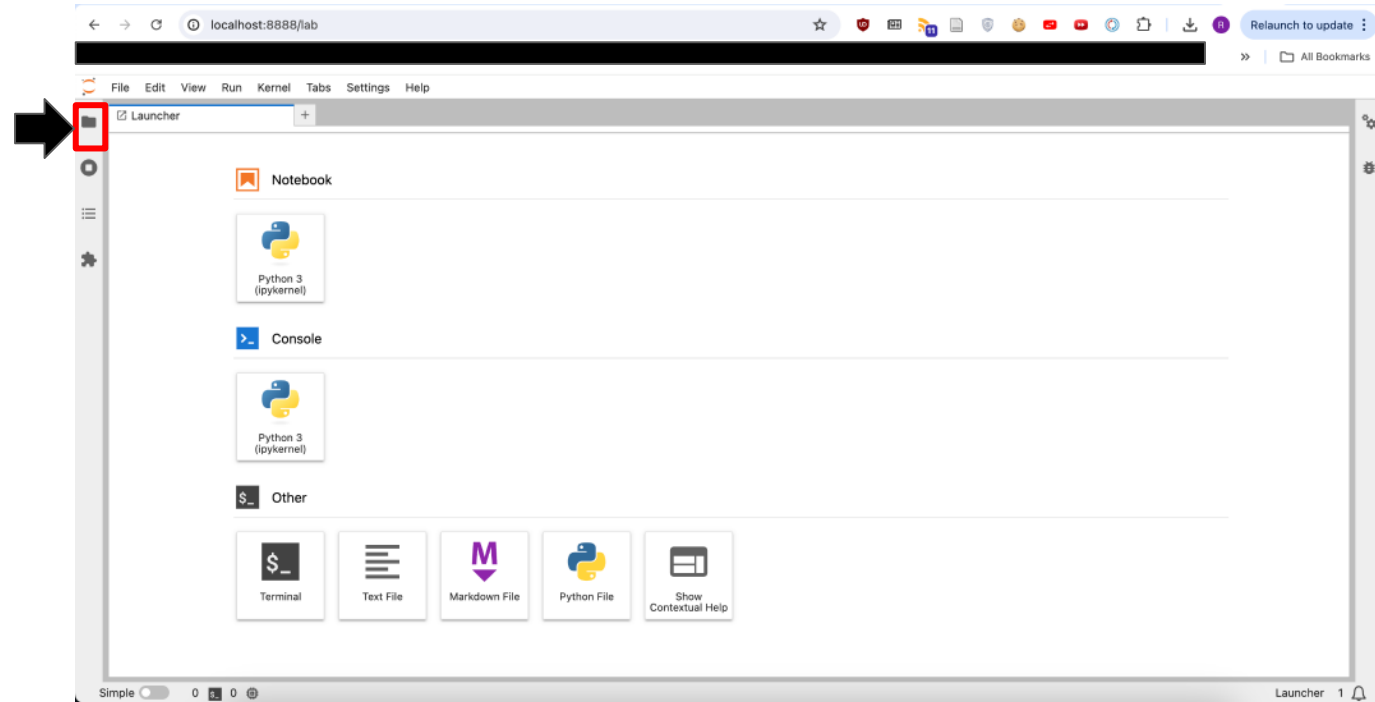
Environment Setup

5) You should see a browser tab that looks like this.



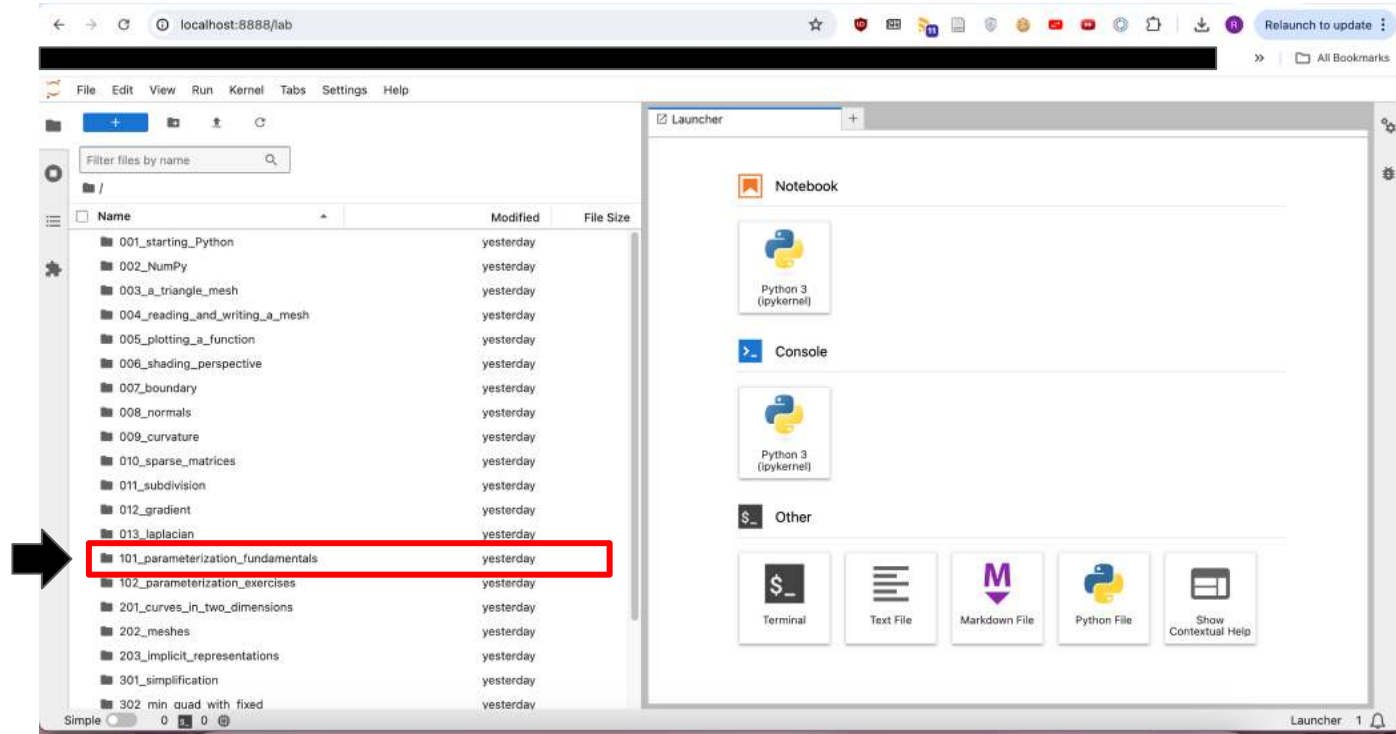
Environment Setup

6) You can access the folders in sgi-introduction-course by clicking the upper left folder icon



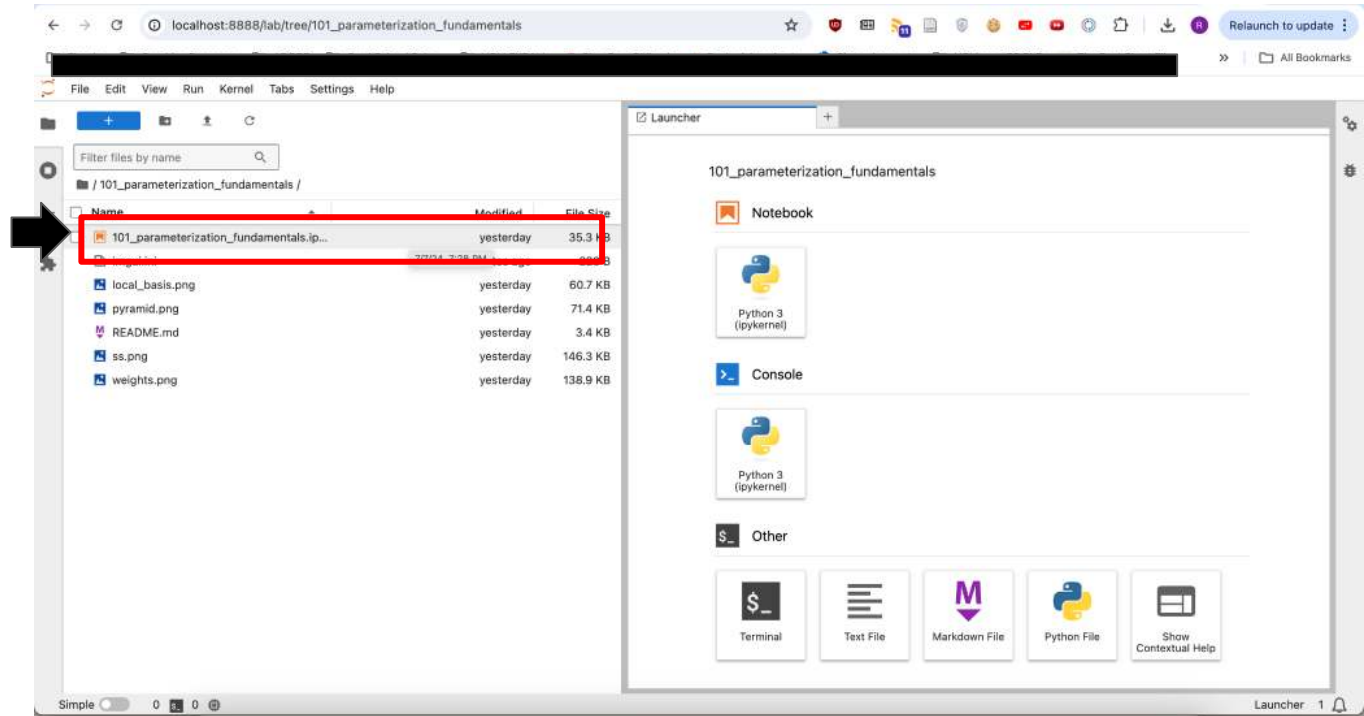
Environment Setup

7) Navigate to 101_parameterization_fundamentals



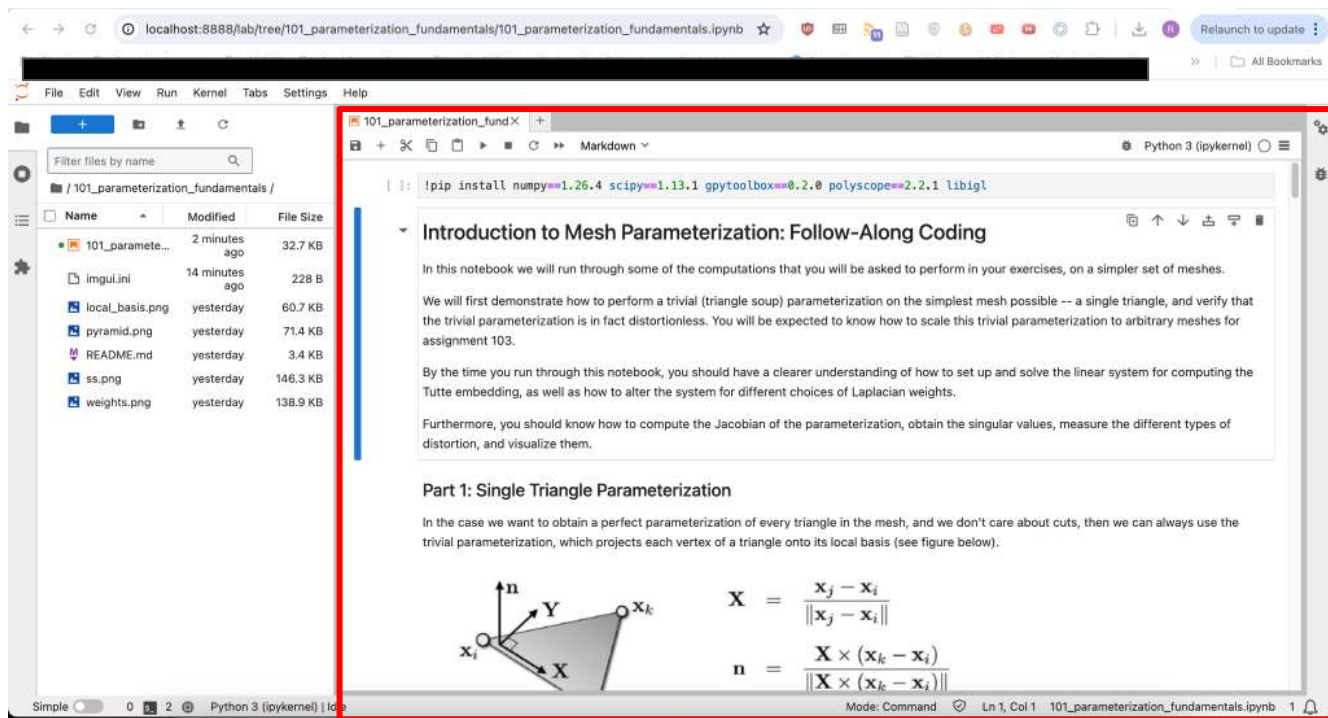
Environment Setup

8) Open 101_parameterization_fundamentals.ipynb



Environment Setup

9) You should now see the notebook open in the right panel.



The screenshot displays a JupyterLab environment. On the left, a file browser shows the directory structure of the '101_parameterization_fundamentals' project, including files like 'imgui.ini', 'local_basis.png', 'pyramid.png', 'README.md', 'ss.png', and 'weights.png'. The right panel shows a Jupyter notebook titled '101_parameterization_fundamentals.ipynb'. The notebook content includes a code cell with pip installation commands for various libraries, followed by an introduction to mesh parameterization and a section on single triangle parameterization. A diagram of a triangle with vertices x_i , x_j , and x_k is shown, along with the formulas for the transformation matrix X and the normal vector n .

```
!pip install numpy==1.26.4 scipy==1.13.1 gpytoolbox==0.2.0 polyscope==2.2.1 libigl
```

Introduction to Mesh Parameterization: Follow-Along Coding

In this notebook we will run through some of the computations that you will be asked to perform in your exercises, on a simpler set of meshes.

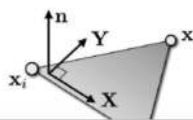
We will first demonstrate how to perform a trivial (triangle soup) parameterization on the simplest mesh possible -- a single triangle, and verify that the trivial parameterization is in fact distortionless. You will be expected to know how to scale this trivial parameterization to arbitrary meshes for assignment 103.

By the time you run through this notebook, you should have a clearer understanding of how to set up and solve the linear system for computing the Tutte embedding, as well as how to alter the system for different choices of Laplacian weights.

Furthermore, you should know how to compute the Jacobian of the parameterization, obtain the singular values, measure the different types of distortion, and visualize them.

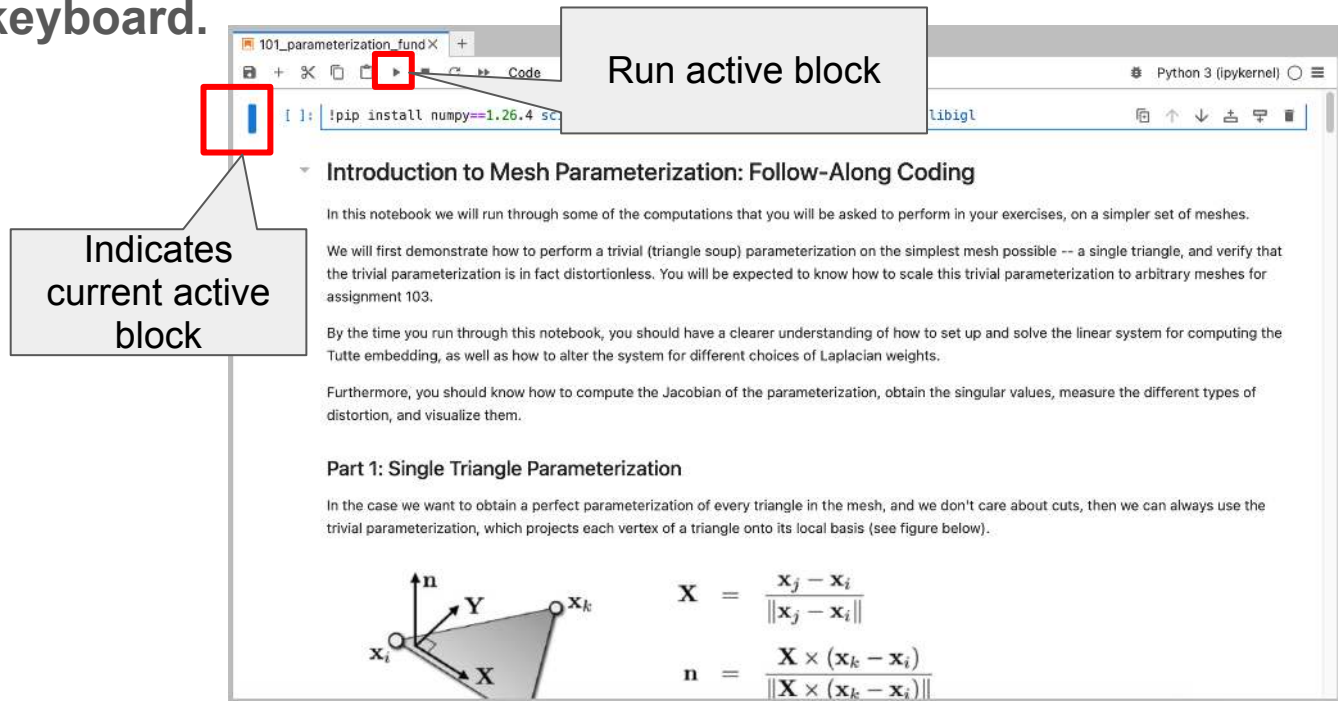
Part 1: Single Triangle Parameterization

In the case we want to obtain a perfect parameterization of every triangle in the mesh, and we don't care about cuts, then we can always use the trivial parameterization, which projects each vertex of a triangle onto its local basis (see figure below).


$$X = \frac{x_j - x_i}{\|x_j - x_i\|}$$
$$n = \frac{X \times (x_k - x_i)}{\|X \times (x_k - x_i)\|}$$

Environment Setup

10) Groups of code are separate into blocks. The blue bar indicates the active block. You can run the active block by pressing the ► button or pressing “shift+enter” on your keyboard.



Run active block

Indicates current active block

101_parameterization_fundX

Python 3 (ipykernel)

libigl

```
[ ]: !pip install numpy==1.26.4 sc
```

Introduction to Mesh Parameterization: Follow-Along Coding

In this notebook we will run through some of the computations that you will be asked to perform in your exercises, on a simpler set of meshes.

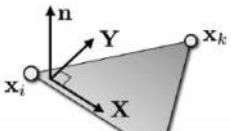
We will first demonstrate how to perform a trivial (triangle soup) parameterization on the simplest mesh possible -- a single triangle, and verify that the trivial parameterization is in fact distortionless. You will be expected to know how to scale this trivial parameterization to arbitrary meshes for assignment 103.

By the time you run through this notebook, you should have a clearer understanding of how to set up and solve the linear system for computing the Tutte embedding, as well as how to alter the system for different choices of Laplacian weights.

Furthermore, you should know how to compute the Jacobian of the parameterization, obtain the singular values, measure the different types of distortion, and visualize them.

Part 1: Single Triangle Parameterization

In the case we want to obtain a perfect parameterization of every triangle in the mesh, and we don't care about cuts, then we can always use the trivial parameterization, which projects each vertex of a triangle onto its local basis (see figure below).


$$\mathbf{X} = \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|}$$
$$\mathbf{n} = \frac{\mathbf{X} \times (\mathbf{x}_k - \mathbf{x}_i)}{\|\mathbf{X} \times (\mathbf{x}_k - \mathbf{x}_i)\|}$$

Environment Setup



```
!pip install numpy==1.26.4 scipy==1.13.1 gpytoolbox==0.2.0 polyscope==2.2.1 libigl
```

Stops the
current code
execution

Introduction to Mesh Parameterization: Follow-Along Coding

In this notebook we will run through some of the computations that you will be asked to perform in your exercises, on a simpler set of meshes.

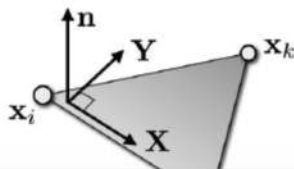
We will first demonstrate how to perform a trivial (triangle soup) parameterization on the simplest mesh possible -- a single triangle, and verify that the trivial parameterization is in fact distortionless. You will be expected to know how to scale this trivial parameterization to arbitrary meshes for assignment 103.

By the time you run through this notebook, you should have a clearer understanding of how to set up and solve the linear system for computing the Tutte embedding, as well as how to alter the system for different choices of Laplacian weights.

Furthermore, you should know how to compute the Jacobian of the parameterization, obtain the singular values, measure the different types of distortion, and visualize them.

Part 1: Single Triangle Parameterization

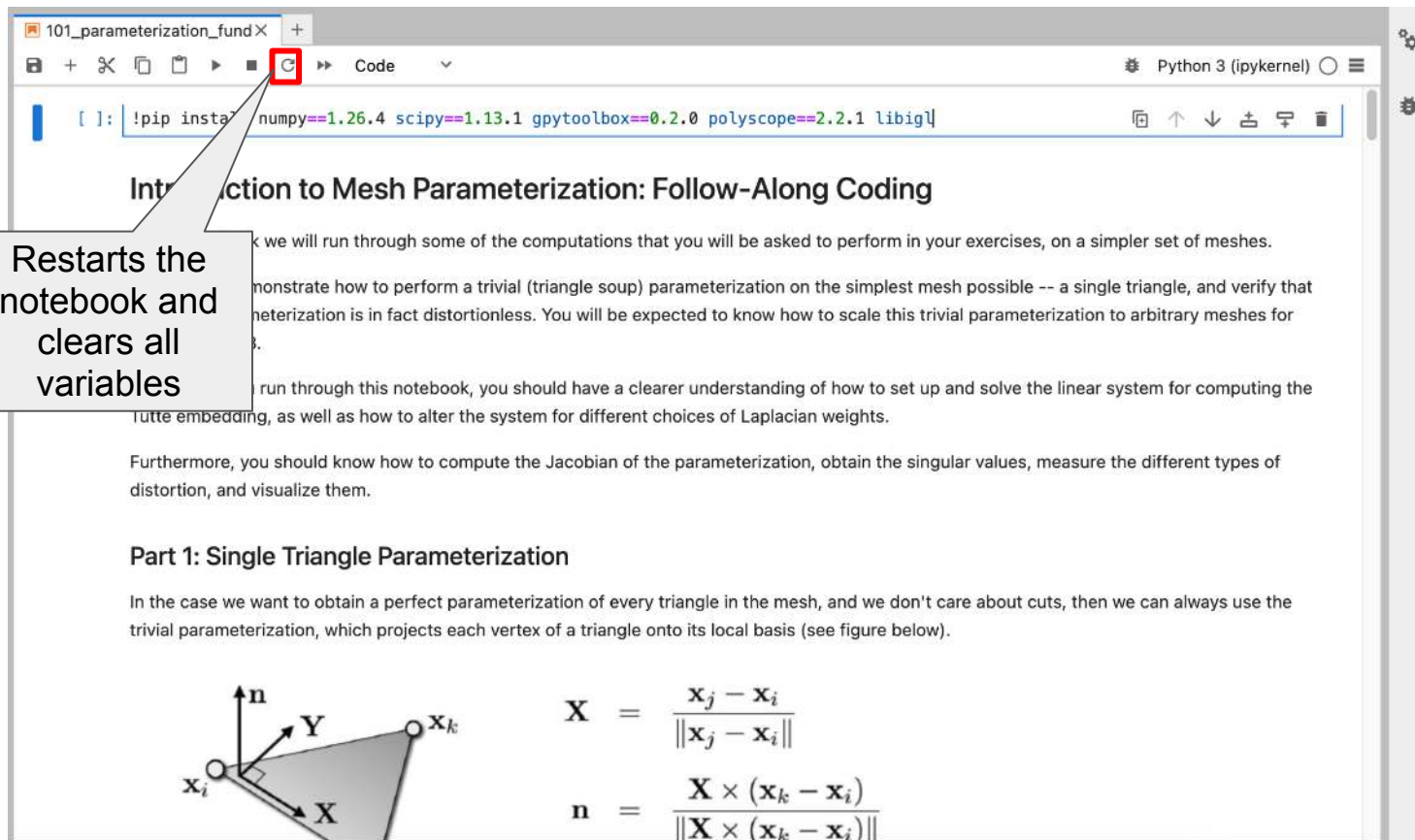
In the case we want to obtain a perfect parameterization of every triangle in the mesh, and we don't care about cuts, then we can always use the trivial parameterization, which projects each vertex of a triangle onto its local basis (see figure below).



$$\mathbf{X} = \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|}$$
$$\mathbf{n} = \frac{\mathbf{X} \times (\mathbf{x}_k - \mathbf{x}_i)}{\|\mathbf{X} \times (\mathbf{x}_k - \mathbf{x}_i)\|}$$

Environment Setup

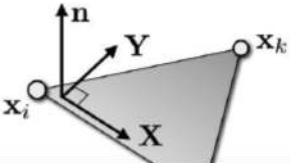
Restarts the notebook and clears all variables



The screenshot shows a Jupyter Notebook window titled "101_parameterization_fund X". The toolbar at the top includes a red square icon with a circular arrow, which is highlighted by a red box and a callout bubble. The callout bubble contains the text "Restarts the notebook and clears all variables". Below the toolbar, the code cell contains the command: `!pip install numpy==1.26.4 scipy==1.13.1 gpytoolbox==0.2.0 polyscope==2.2.1 libigl`. The notebook content includes the title "Introduction to Mesh Parameterization: Follow-Along Coding" and several paragraphs of text. The text describes the goal of the notebook: to run through some of the computations that you will be asked to perform in your exercises, on a simpler set of meshes. It mentions demonstrating how to perform a trivial (triangle soup) parameterization on the simplest mesh possible -- a single triangle, and verifying that the parameterization is in fact distortionless. It also mentions that you will be expected to know how to scale this trivial parameterization to arbitrary meshes for more complex cases. The text further states that by running through this notebook, you should have a clearer understanding of how to set up and solve the linear system for computing the Tutte embedding, as well as how to alter the system for different choices of Laplacian weights. Additionally, it mentions that you should know how to compute the Jacobian of the parameterization, obtain the singular values, measure the different types of distortion, and visualize them.

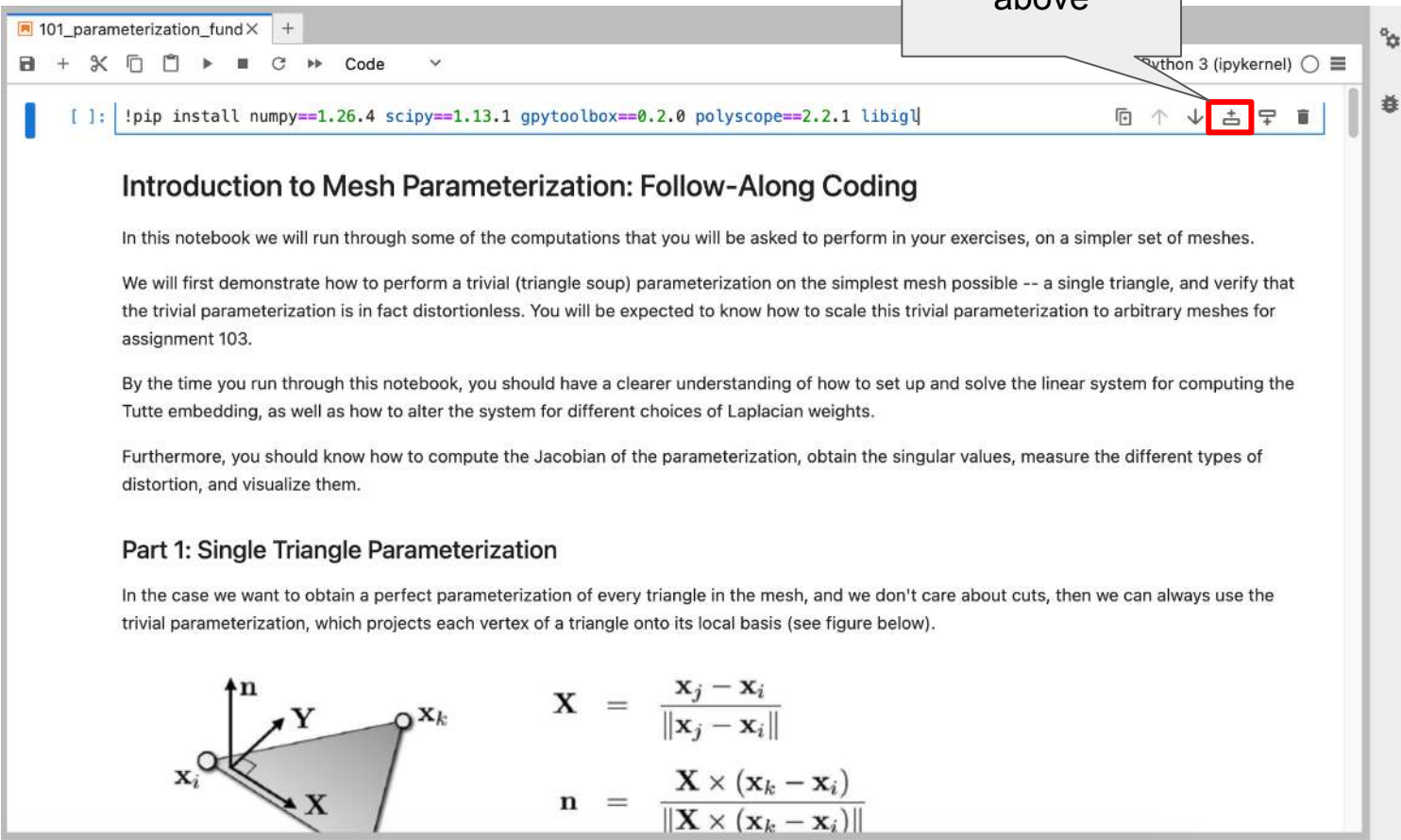
Part 1: Single Triangle Parameterization

In the case we want to obtain a perfect parameterization of every triangle in the mesh, and we don't care about cuts, then we can always use the trivial parameterization, which projects each vertex of a triangle onto its local basis (see figure below).


$$\mathbf{X} = \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|}$$
$$\mathbf{n} = \frac{\mathbf{X} \times (\mathbf{x}_k - \mathbf{x}_i)}{\|\mathbf{X} \times (\mathbf{x}_k - \mathbf{x}_i)\|}$$

Environment Setup

Add block
above



The screenshot shows a Jupyter Notebook window titled "101_parameterization_fundX". The toolbar at the top includes icons for saving, adding new cells, copying, pasting, running, and other standard Jupyter actions. A callout box points to the "Add block above" button (represented by a plus sign in a square) in the toolbar. The code cell contains the following command:

```
[ ]: !pip install numpy==1.26.4 scipy==1.13.1 gpytoolbox==0.2.0 polyscope==2.2.1 libigl
```

The notebook content includes the following sections:

Introduction to Mesh Parameterization: Follow-Along Coding

In this notebook we will run through some of the computations that you will be asked to perform in your exercises, on a simpler set of meshes.

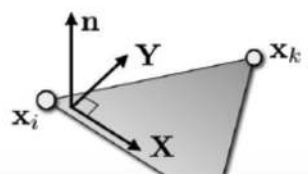
We will first demonstrate how to perform a trivial (triangle soup) parameterization on the simplest mesh possible -- a single triangle, and verify that the trivial parameterization is in fact distortionless. You will be expected to know how to scale this trivial parameterization to arbitrary meshes for assignment 103.

By the time you run through this notebook, you should have a clearer understanding of how to set up and solve the linear system for computing the Tutte embedding, as well as how to alter the system for different choices of Laplacian weights.

Furthermore, you should know how to compute the Jacobian of the parameterization, obtain the singular values, measure the different types of distortion, and visualize them.

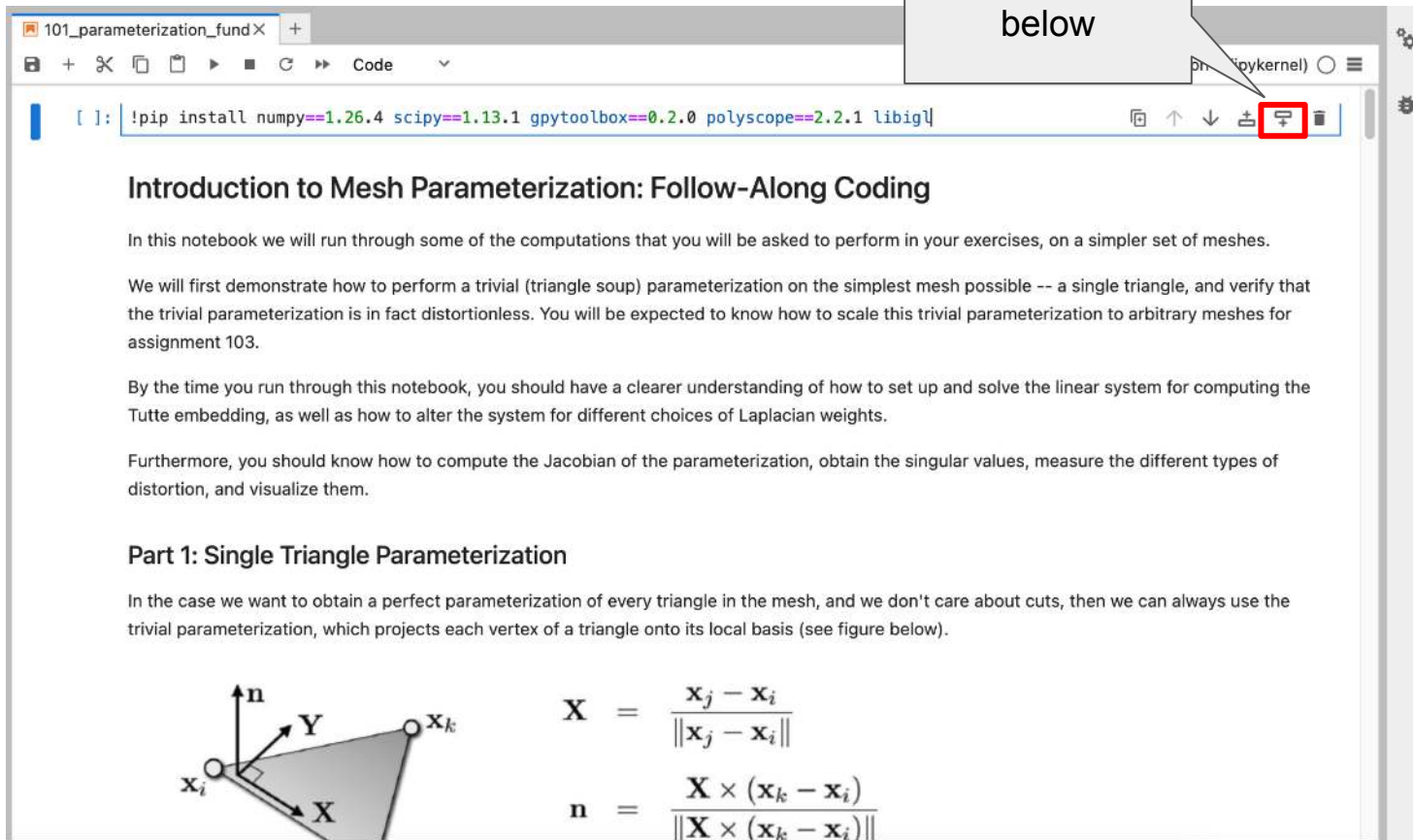
Part 1: Single Triangle Parameterization

In the case we want to obtain a perfect parameterization of every triangle in the mesh, and we don't care about cuts, then we can always use the trivial parameterization, which projects each vertex of a triangle onto its local basis (see figure below).


$$\mathbf{X} = \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|}$$
$$\mathbf{n} = \frac{\mathbf{X} \times (\mathbf{x}_k - \mathbf{x}_i)}{\|\mathbf{X} \times (\mathbf{x}_k - \mathbf{x}_i)\|}$$

Environment Setup

Add block
below



The screenshot shows a Jupyter Notebook window titled "101_parameterization_fund X". The top toolbar includes icons for file operations, execution, and a red-outlined "Add block below" button. The code cell contains the command: `!pip install numpy==1.26.4 scipy==1.13.1 gpytoolbox==0.2.0 polyscope==2.2.1 libigl`. The notebook content includes a title "Introduction to Mesh Parameterization: Follow-Along Coding", an introductory paragraph, and a section titled "Part 1: Single Triangle Parameterization" with a diagram of a triangle and its local basis vectors.

Introduction to Mesh Parameterization: Follow-Along Coding

In this notebook we will run through some of the computations that you will be asked to perform in your exercises, on a simpler set of meshes.

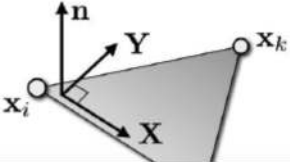
We will first demonstrate how to perform a trivial (triangle soup) parameterization on the simplest mesh possible -- a single triangle, and verify that the trivial parameterization is in fact distortionless. You will be expected to know how to scale this trivial parameterization to arbitrary meshes for assignment 103.

By the time you run through this notebook, you should have a clearer understanding of how to set up and solve the linear system for computing the Tutte embedding, as well as how to alter the system for different choices of Laplacian weights.

Furthermore, you should know how to compute the Jacobian of the parameterization, obtain the singular values, measure the different types of distortion, and visualize them.

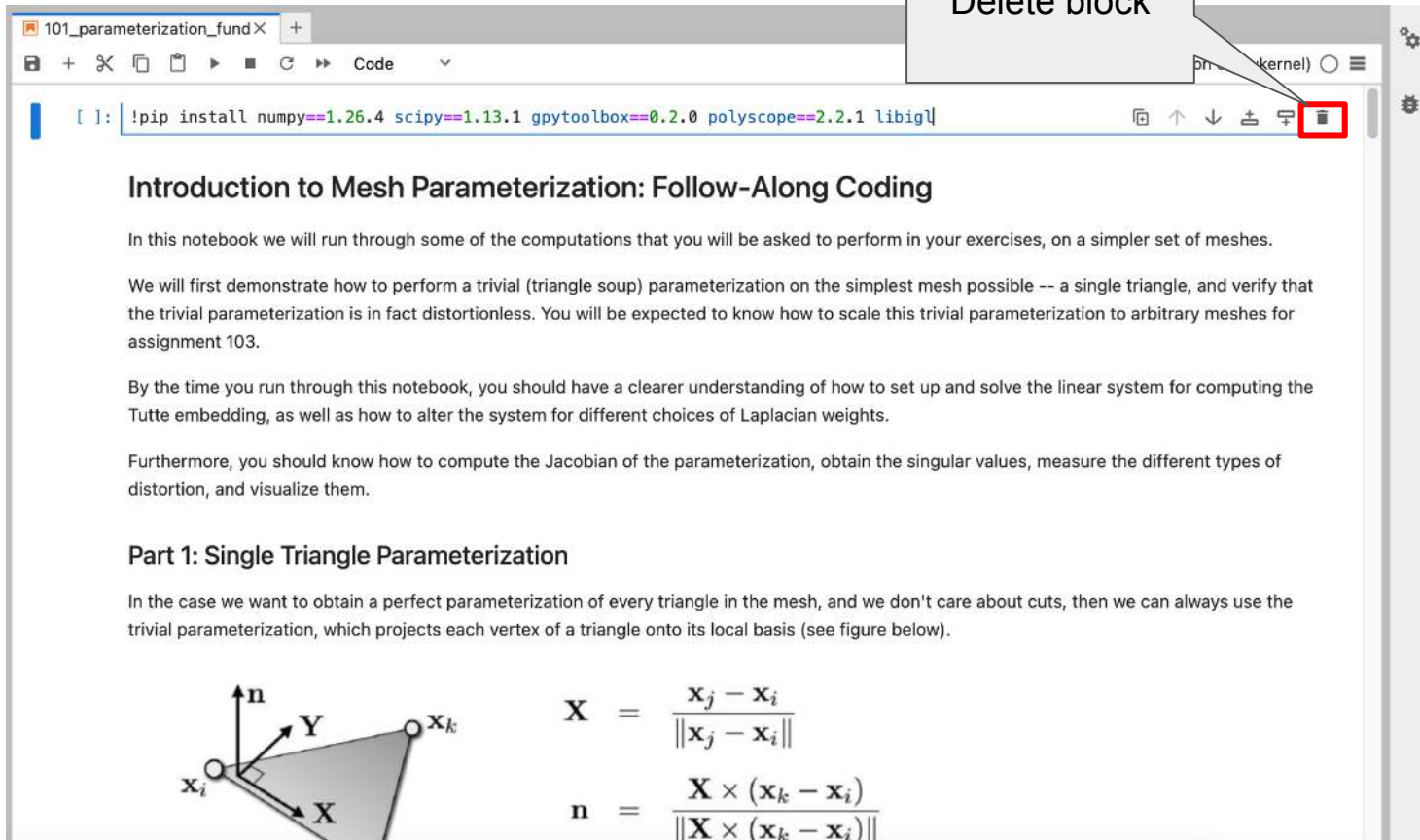
Part 1: Single Triangle Parameterization

In the case we want to obtain a perfect parameterization of every triangle in the mesh, and we don't care about cuts, then we can always use the trivial parameterization, which projects each vertex of a triangle onto its local basis (see figure below).


$$\mathbf{X} = \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|}$$
$$\mathbf{n} = \frac{\mathbf{X} \times (\mathbf{x}_k - \mathbf{x}_i)}{\|\mathbf{X} \times (\mathbf{x}_k - \mathbf{x}_i)\|}$$

Environment Setup

Delete block



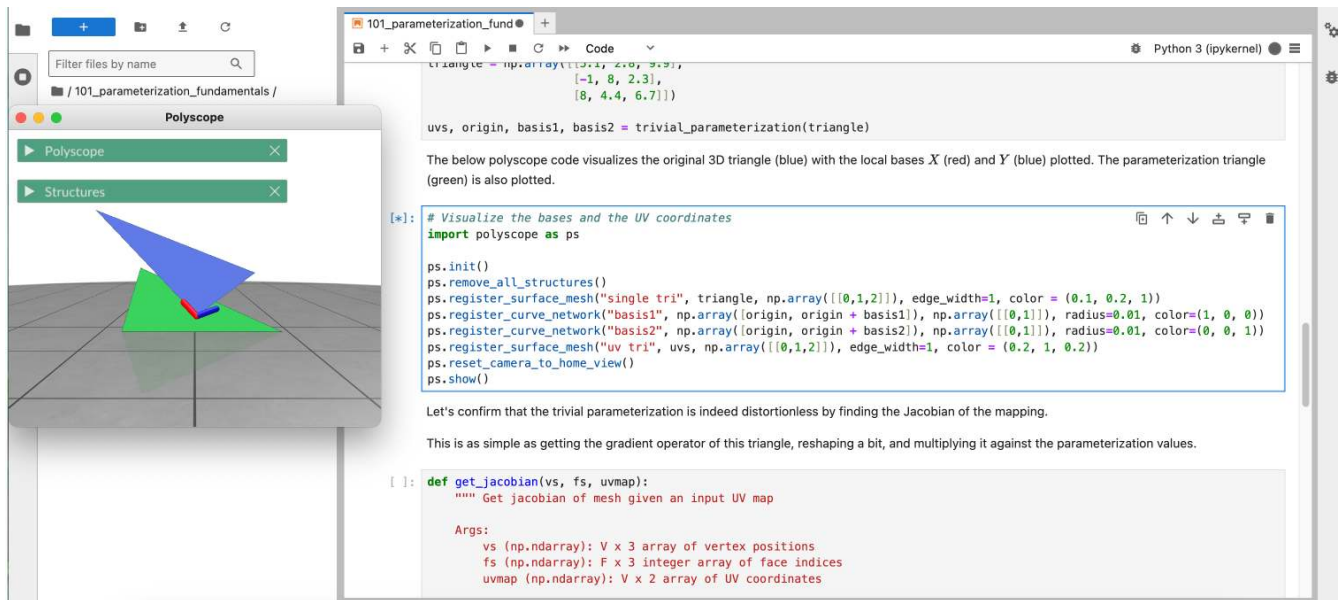
The screenshot shows a Jupyter Notebook interface. The top bar indicates the notebook is named '101_parameterization_fund'. Below the top bar, a code cell contains the command: `!pip install numpy==1.26.4 scipy==1.13.1 gpytoolbox==0.2.0 polyscope==2.2.1 libigl`. To the right of the code cell, a toolbar contains icons for copy, paste, undo, redo, and a trash can icon, which is highlighted with a red box. A callout box labeled 'Delete block' points to this trash can icon. Below the code cell, the notebook content is displayed, featuring the title 'Introduction to Mesh Parameterization: Follow-Along Coding' and several paragraphs of text. The text describes the purpose of the notebook, which is to demonstrate how to perform a trivial (triangle soup) parameterization on the simplest mesh possible -- a single triangle, and verify that the trivial parameterization is in fact distortionless. It also mentions that by the time you run through this notebook, you should have a clearer understanding of how to set up and solve the linear system for computing the Tutte embedding, as well as how to alter the system for different choices of Laplacian weights. Furthermore, it states that you should know how to compute the Jacobian of the parameterization, obtain the singular values, measure the different types of distortion, and visualize them. Below the text, the section 'Part 1: Single Triangle Parameterization' is introduced. The text explains that in the case we want to obtain a perfect parameterization of every triangle in the mesh, and we don't care about cuts, then we can always use the trivial parameterization, which projects each vertex of a triangle onto its local basis (see figure below). The figure shows a triangle with vertices x_i , x_j , and x_k . A local basis is defined by vectors X and Y originating from x_i . The normal vector n is shown perpendicular to the plane of the triangle. The equations for X and n are provided:

$$X = \frac{x_j - x_i}{\|x_j - x_i\|}$$
$$n = \frac{X \times (x_k - x_i)}{\|X \times (x_k - x_i)\|}$$

Environment Setup

101 is designed for you to be able to run the code all the way without any additional coding.

Polyscope will work when run from a cell!



Thank you!