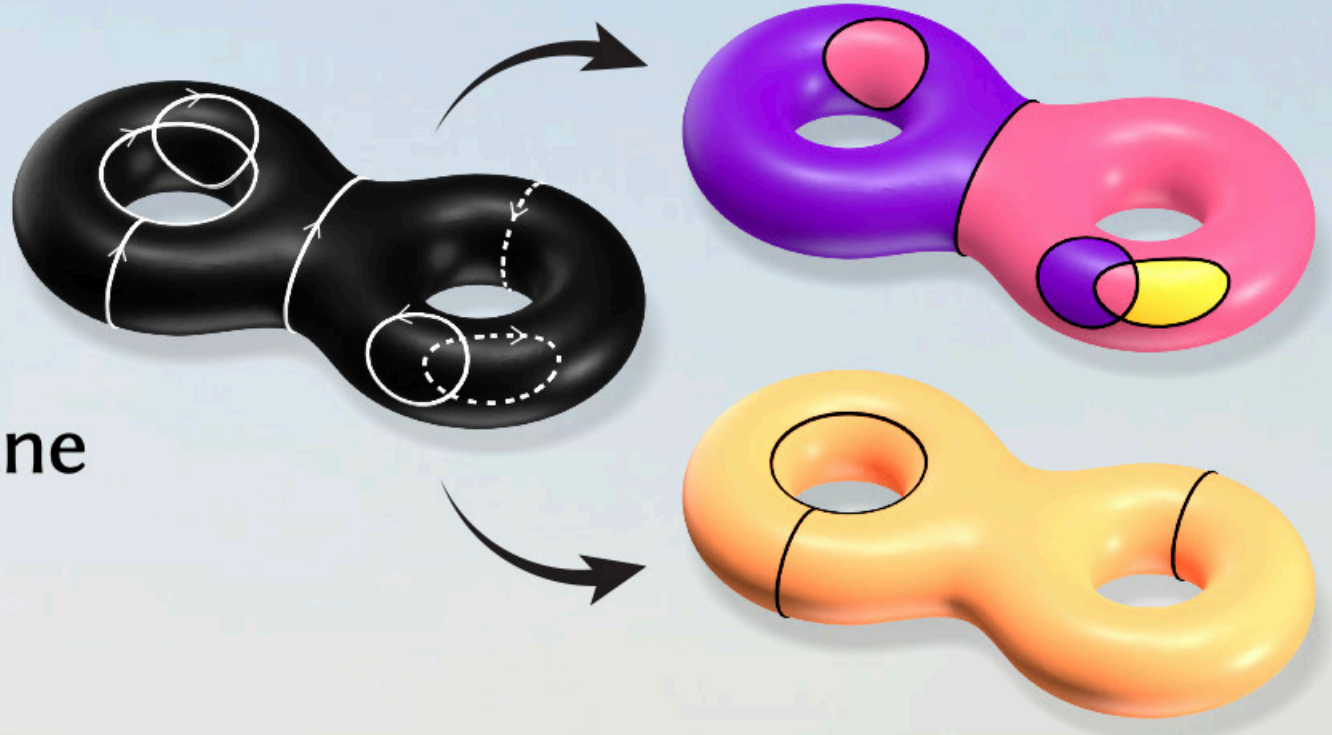
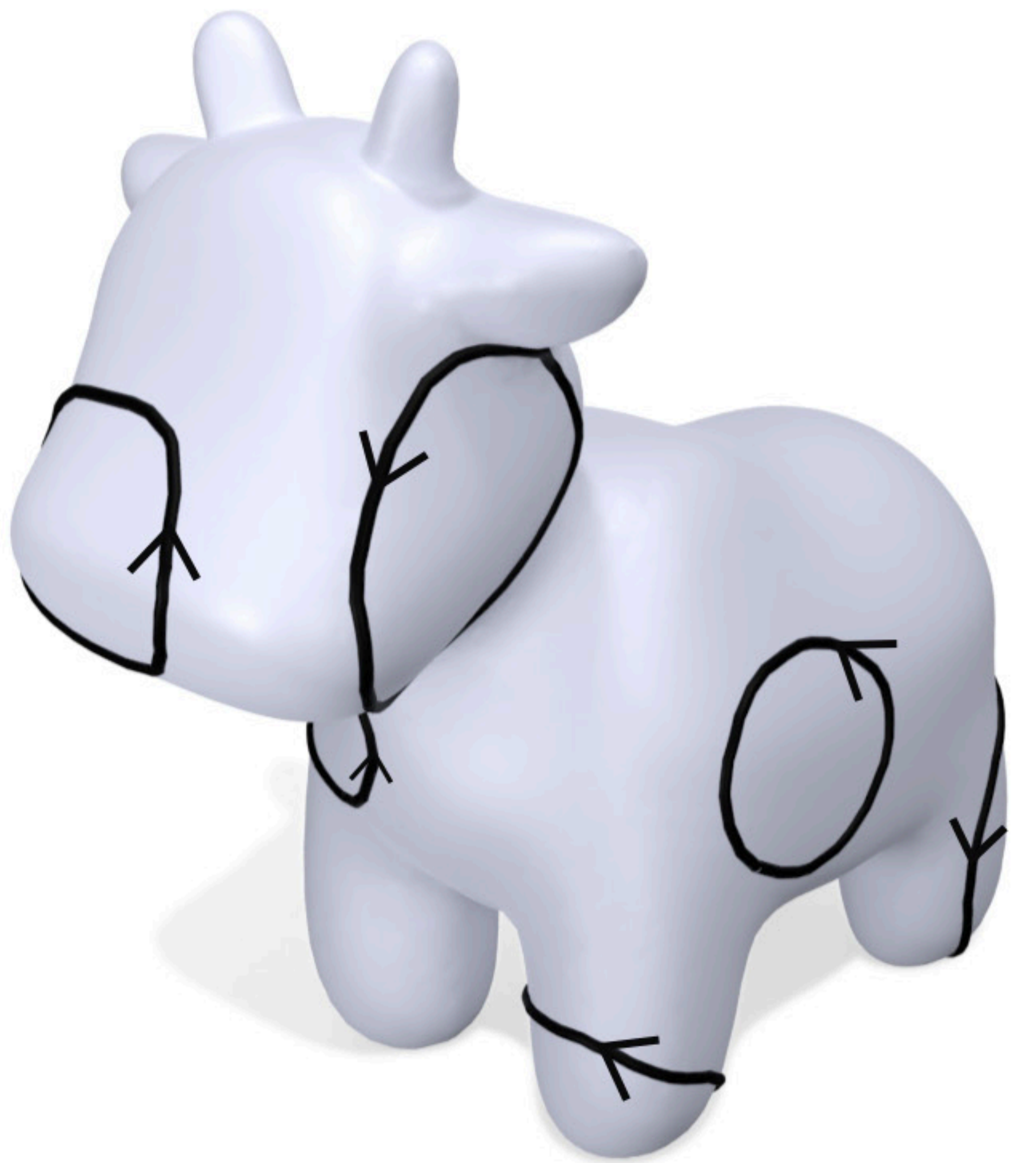
Winding Numbers on Discrete Surfaces

Nicole Feng, Mark Gillespie, Keenan Crane
Carnegie Mellon University

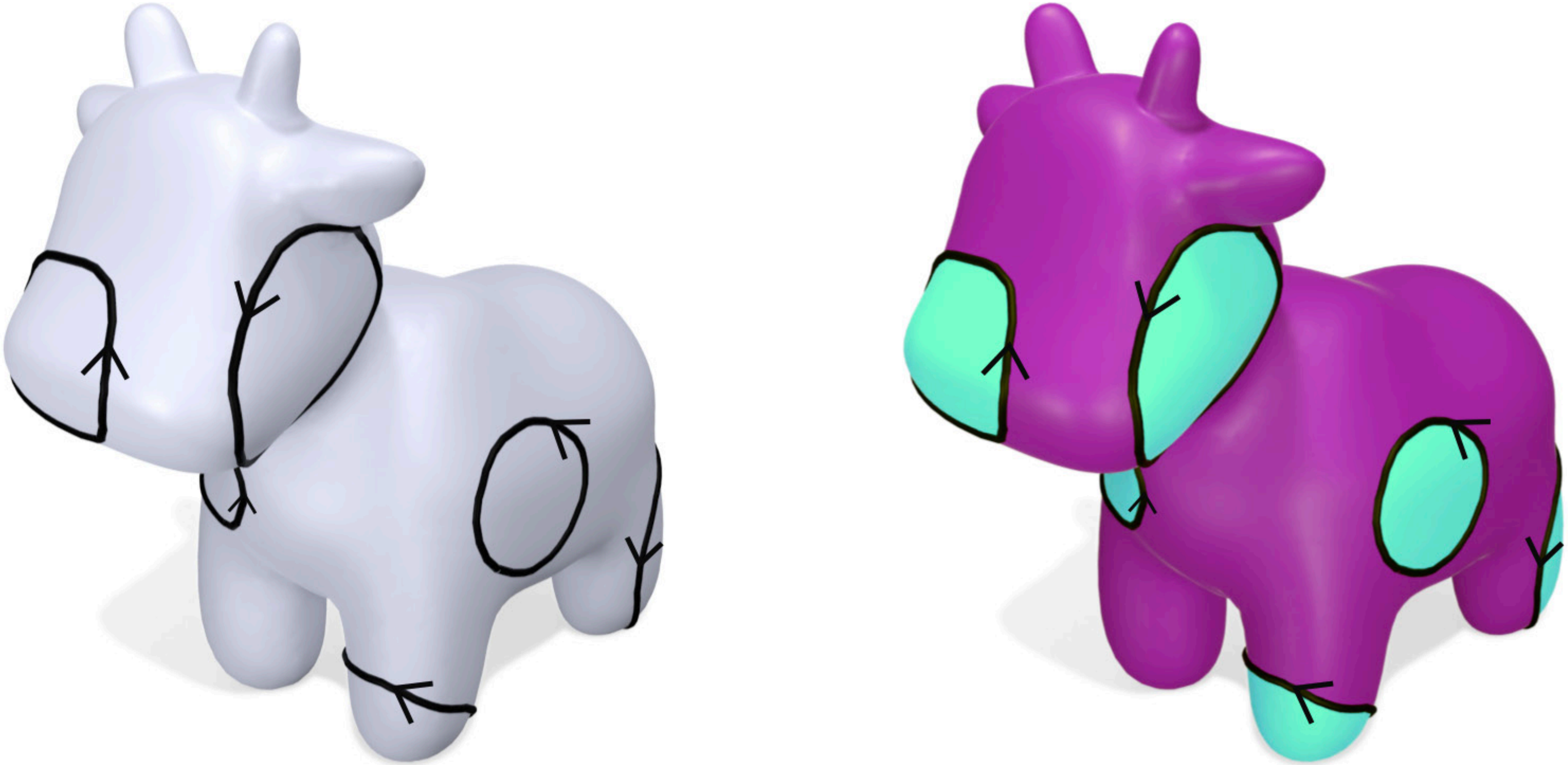


[date]

Problem



Problem



Not all curves bound regions!

Not all curves bound regions!

Fence bounds a region:

Not all curves bound regions!

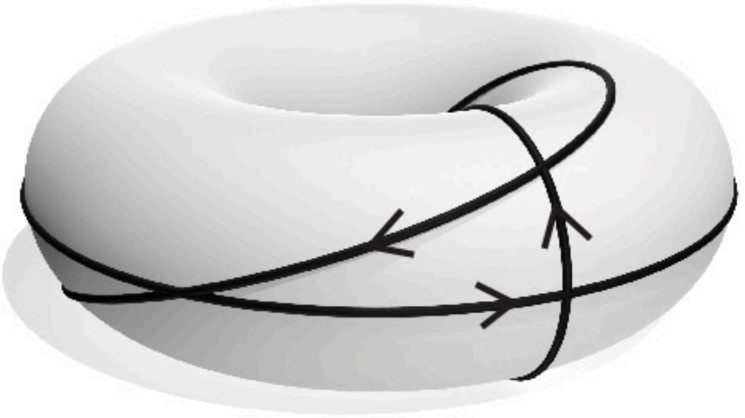
Fence bounds a region:

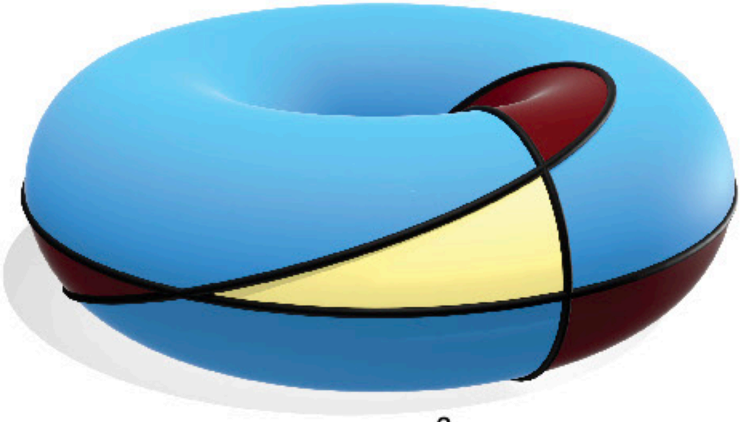
Fence doesn't bound a region:

Not all curves bound regions!

Fence bounds a region:

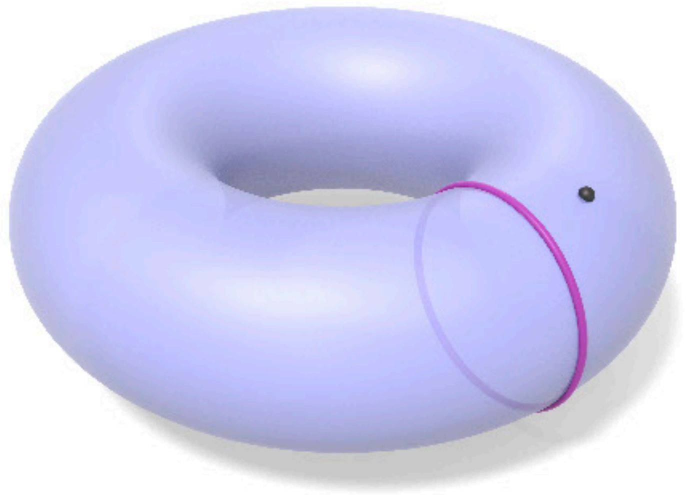
Fence doesn’t bound a region:





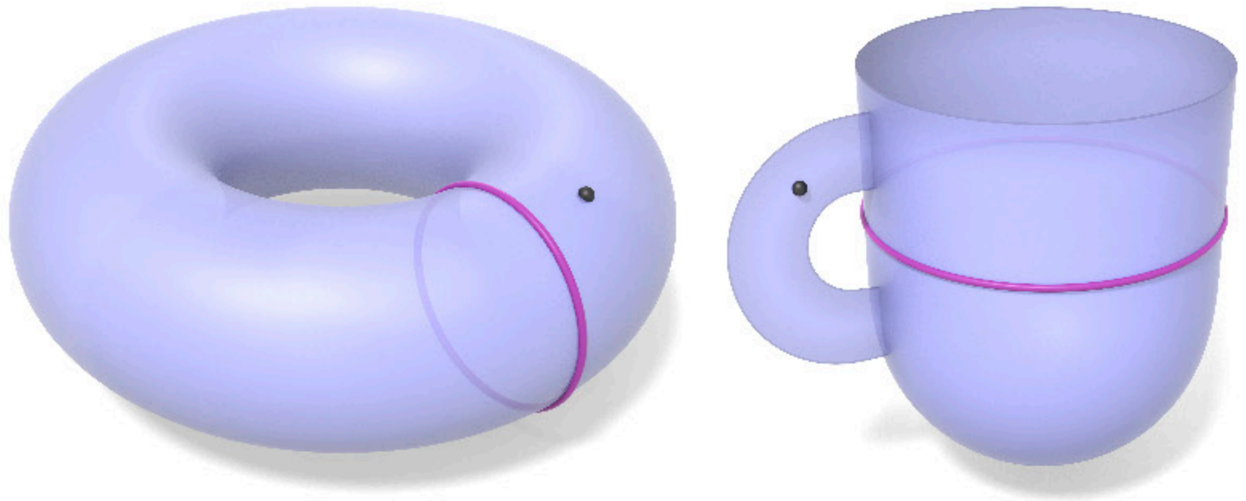
Not as simple as classifying loops

Is the point inside or outside the curve?

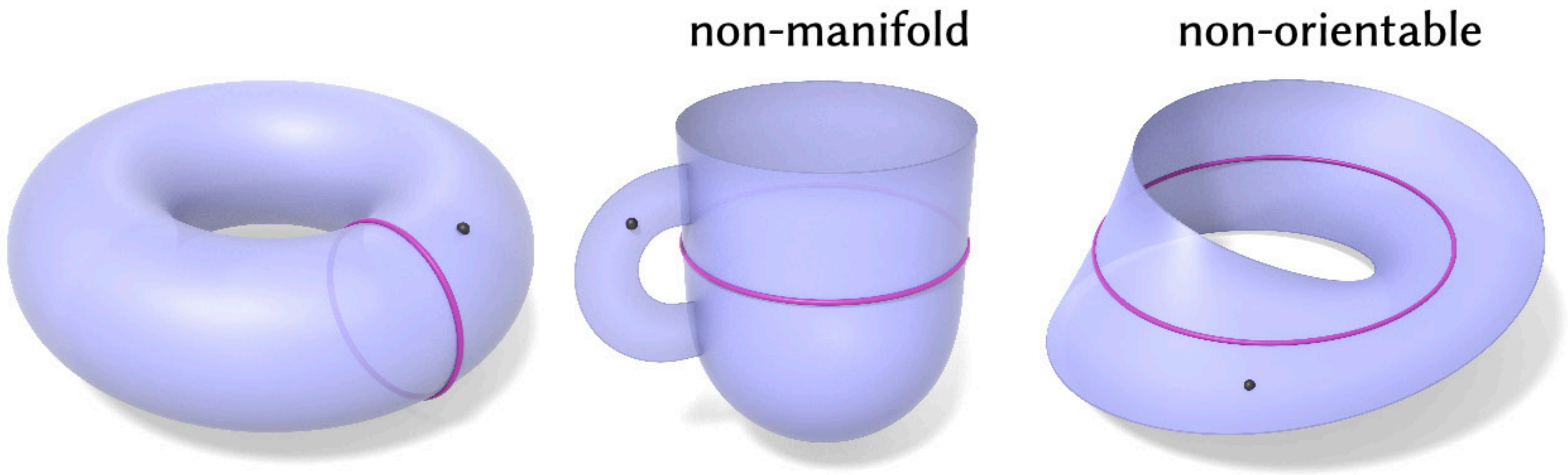


Is the point inside or outside the curve?

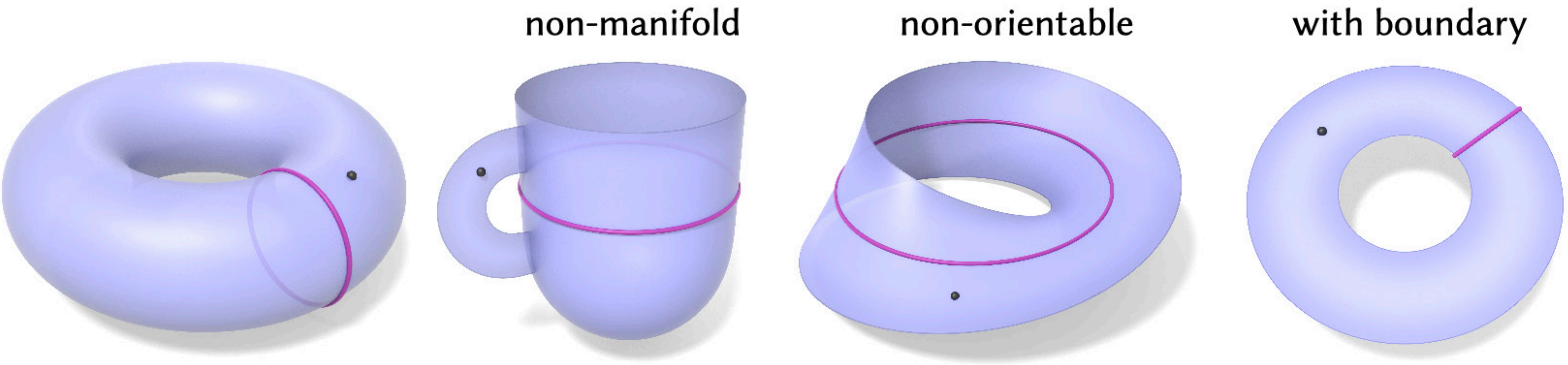
non-manifold



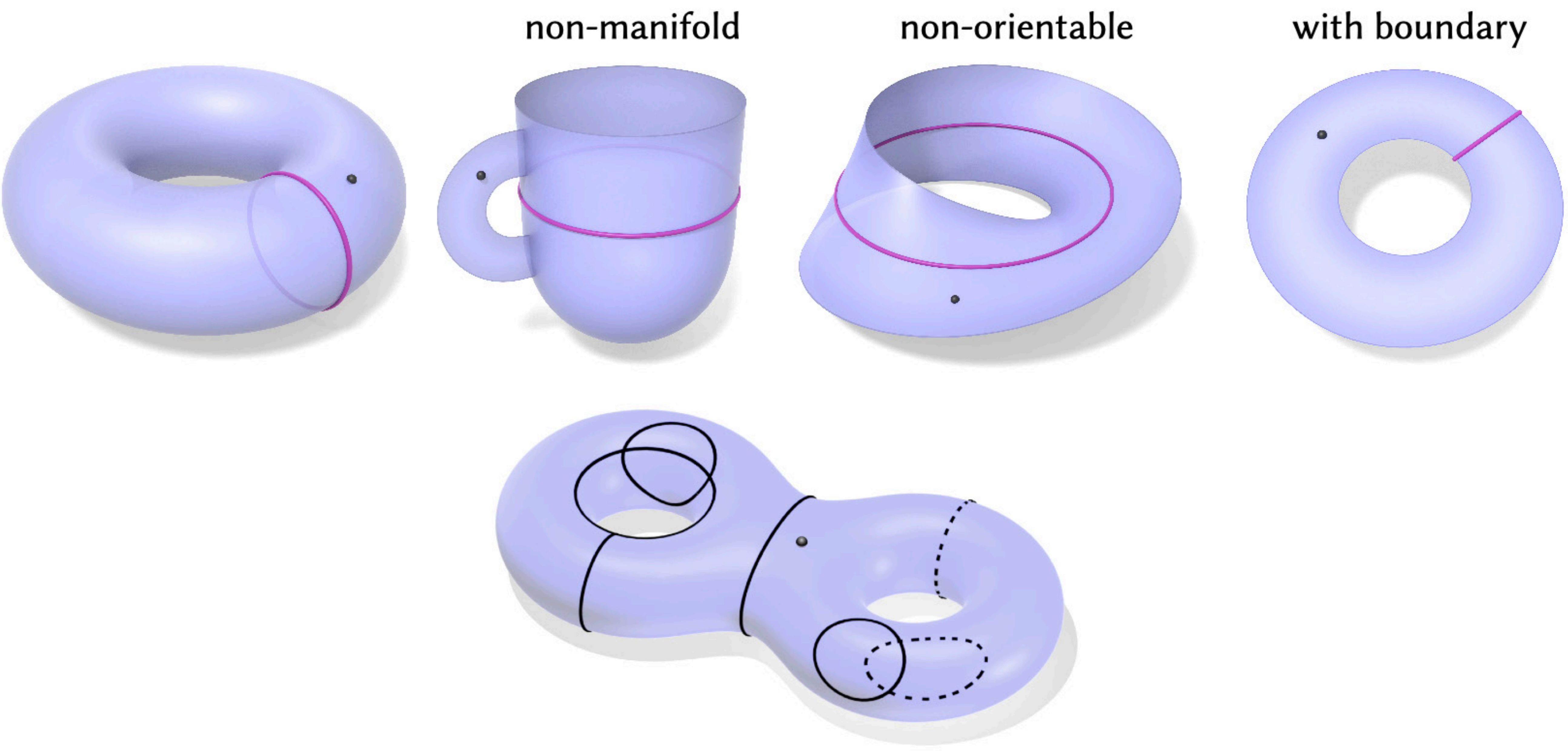
Is the point inside or outside the curve?



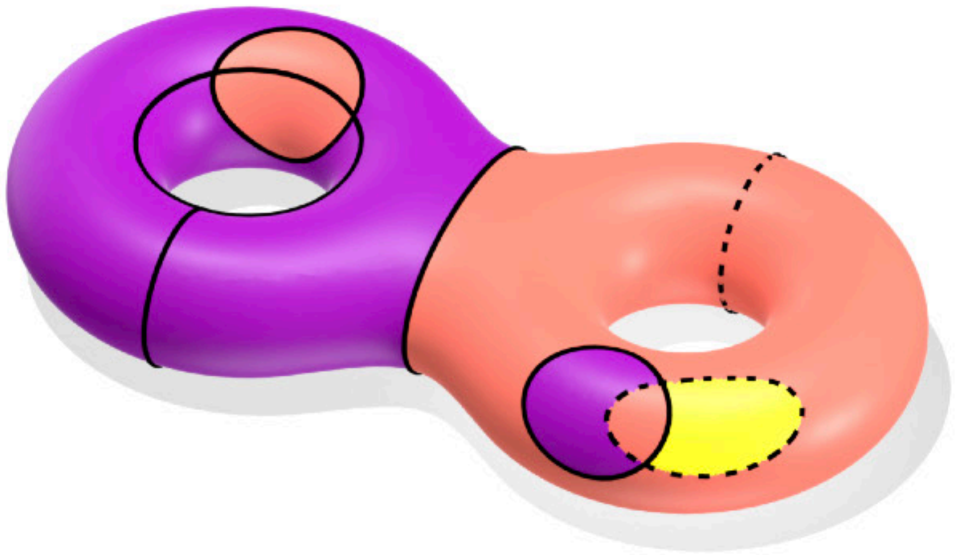
Is the point inside or outside the curve?

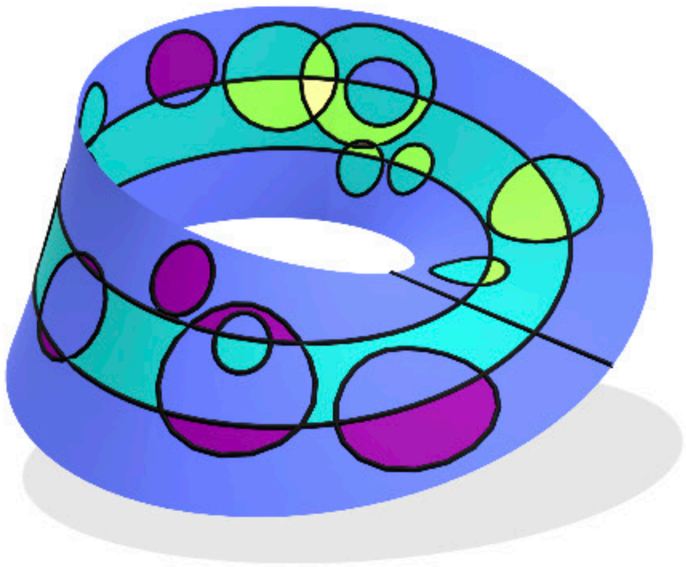


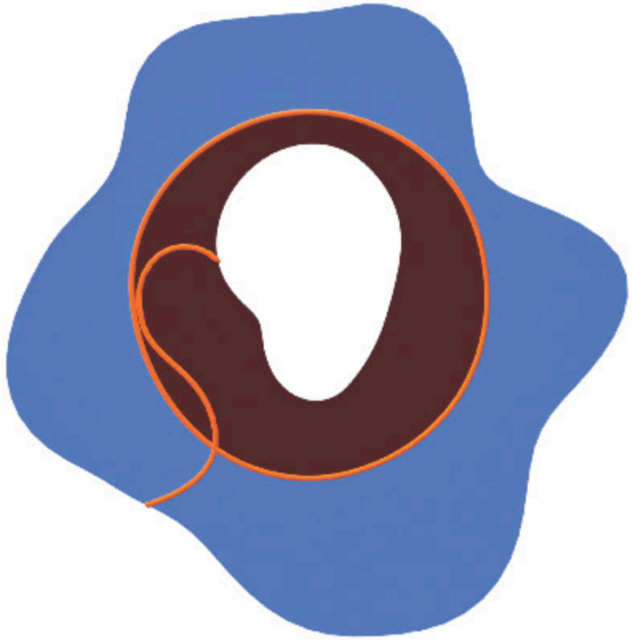
Is the point inside or outside the curve?



Surface Winding Numbers (SWN)

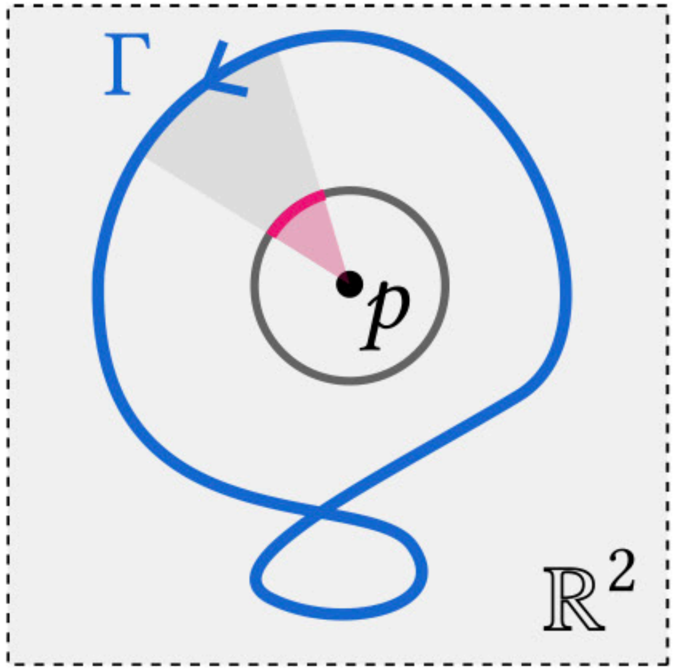




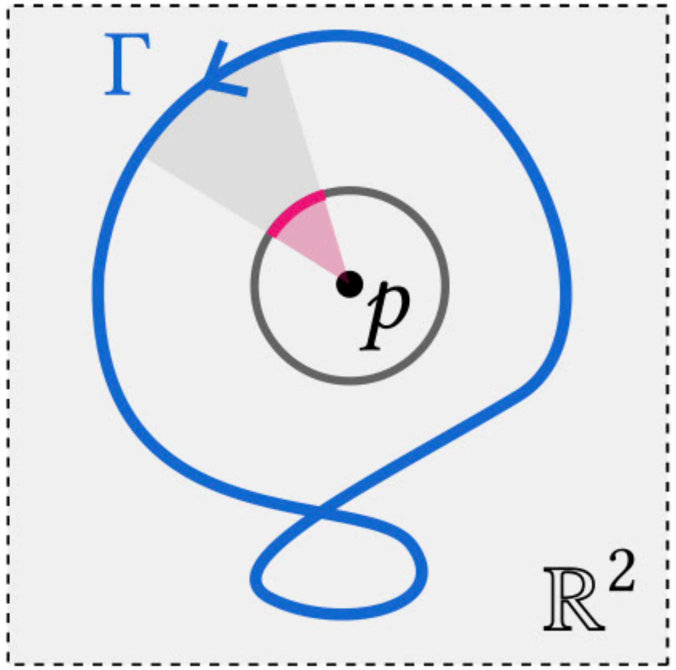


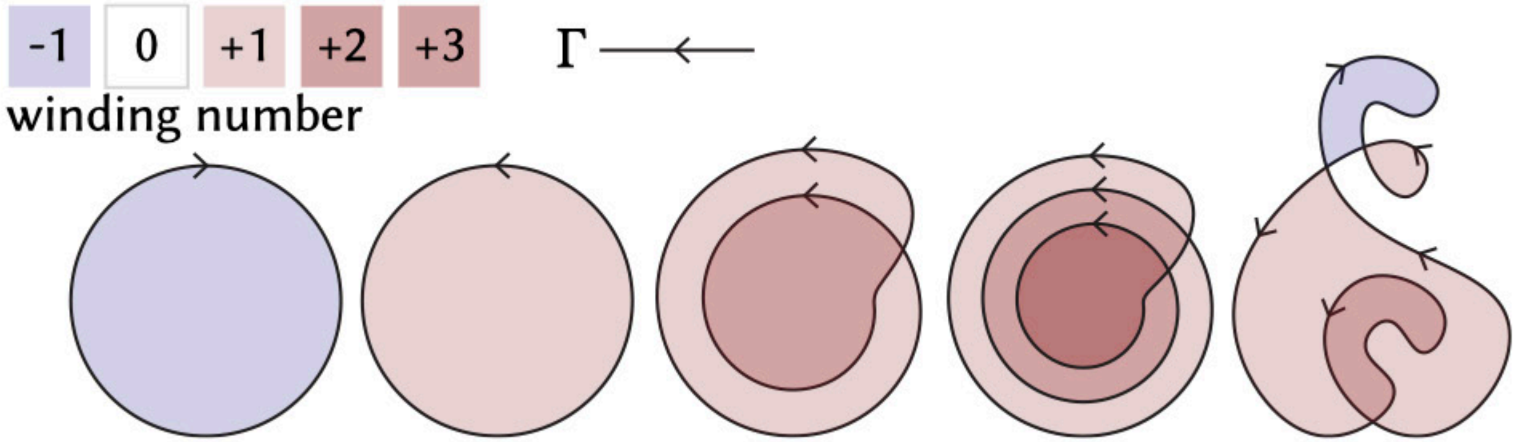
Winding number (solid angle)

Winding number (solid angle)

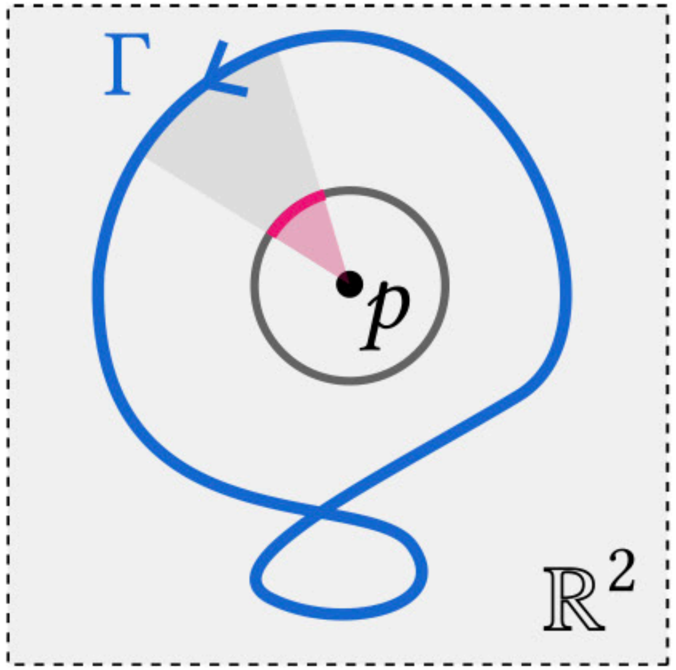


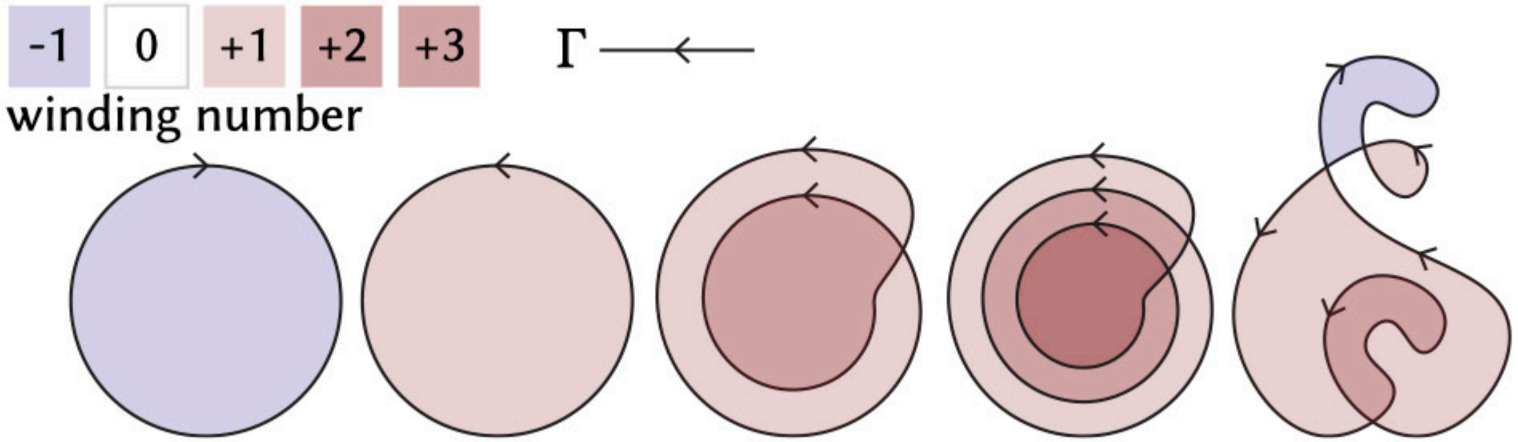
Winding number (solid angle)





Winding number (solid angle)



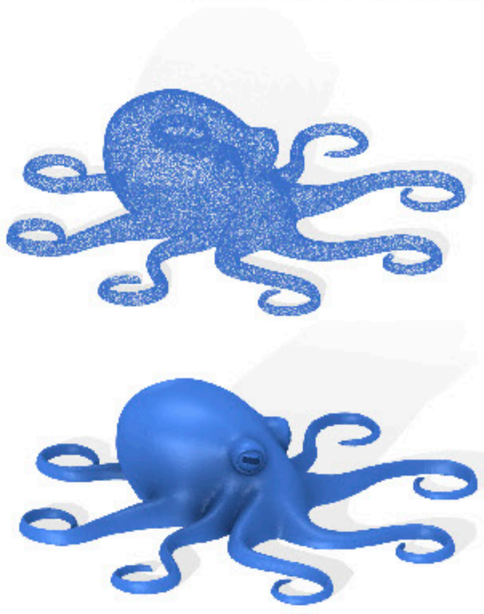


Complex analysis, differential geometry, topology, electromagnetism...

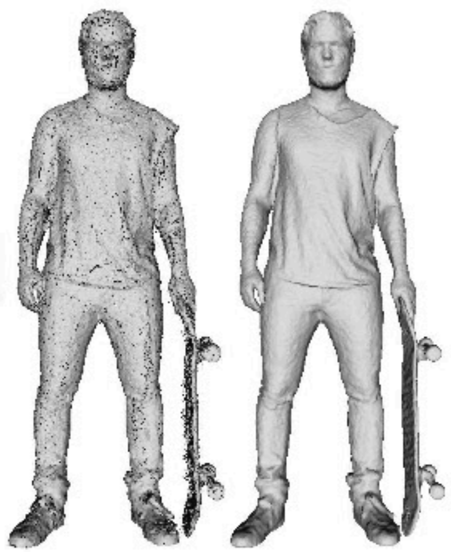
Winding numbers have succeeded before!

Winding numbers have succeeded before!

surface reconstruction

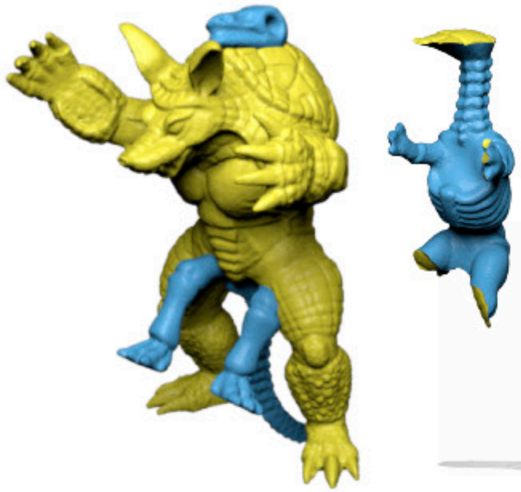


[Barill et al. 2018]

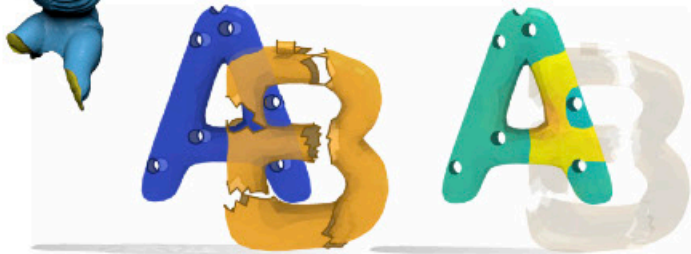


[Collet et al. 2015]

mesh booleans



[Zhou et al. 2016]



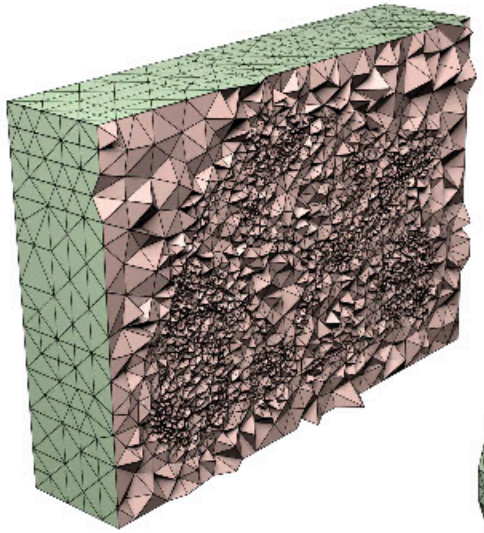
[Barill et al. 2018]

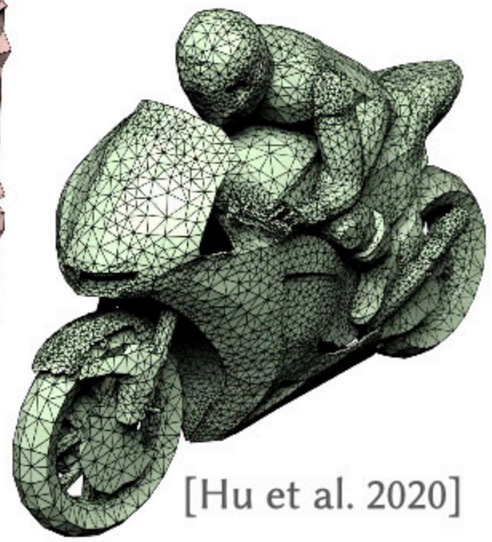
iterative normal estimation

[Xu et al. 2023]

[Hou et al. 2022]

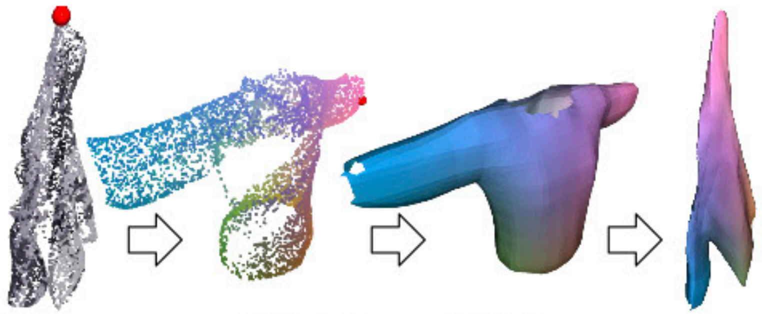
tet-meshing



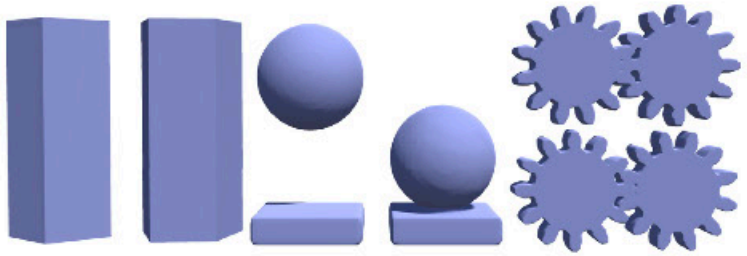


[Hu et al. 2020]

geometric preprocessing



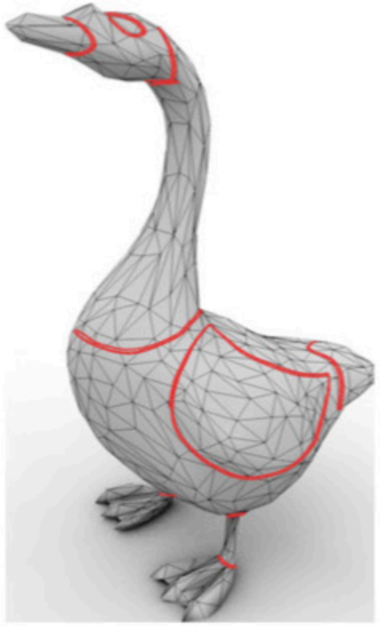
[Chi, Song 2021]



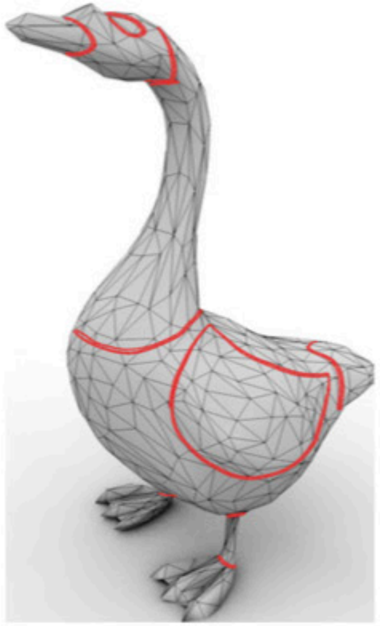
[Dvořák et al. 2021]

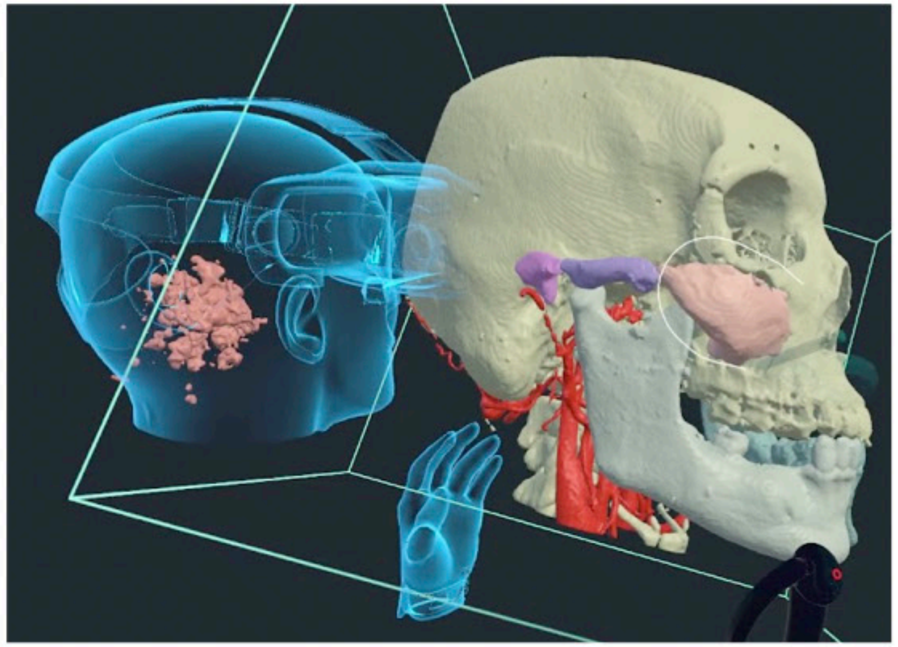
Our motivation

Our motivation

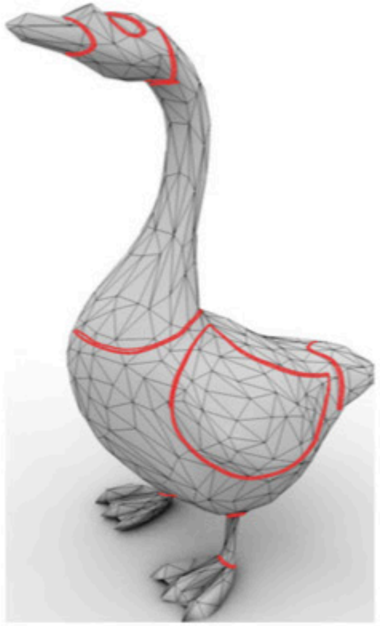


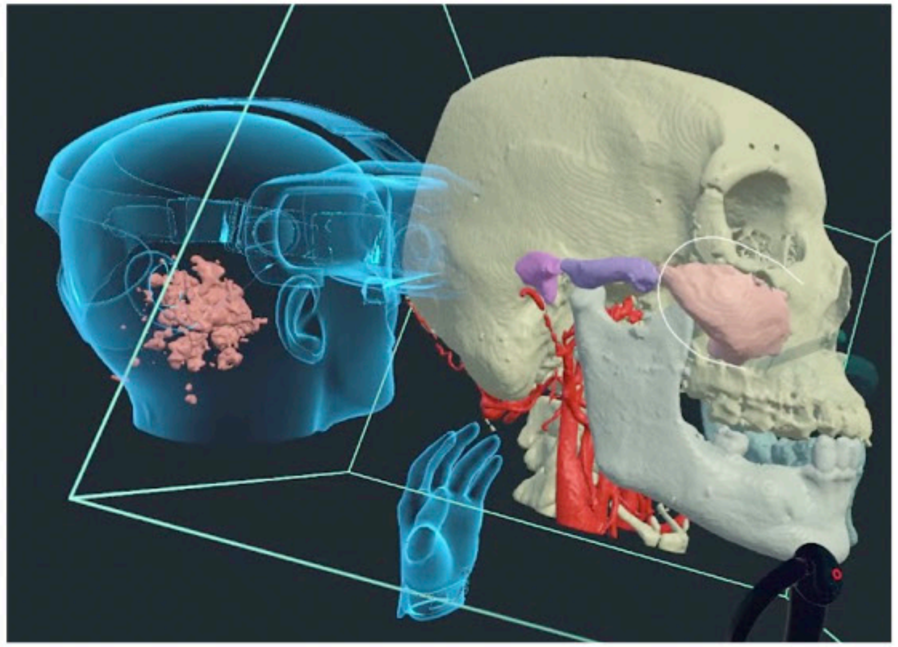
Our motivation

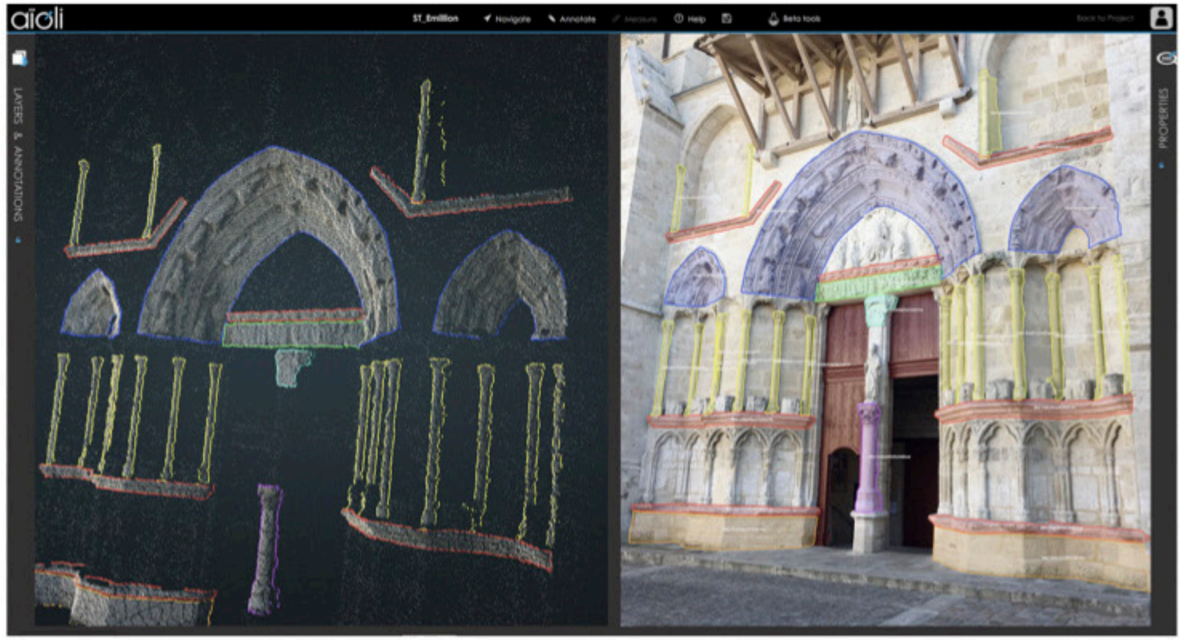


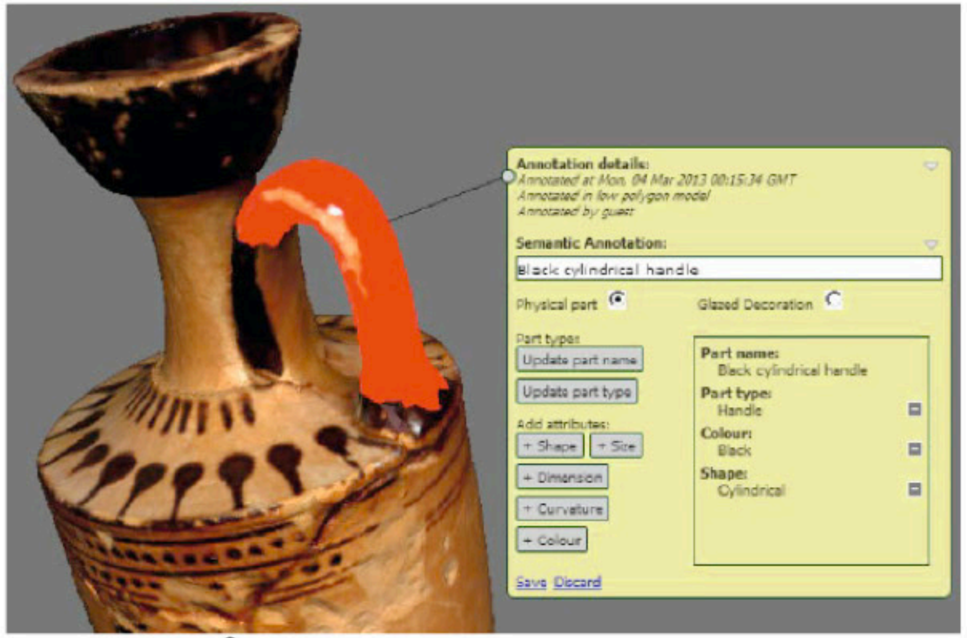


Our motivation

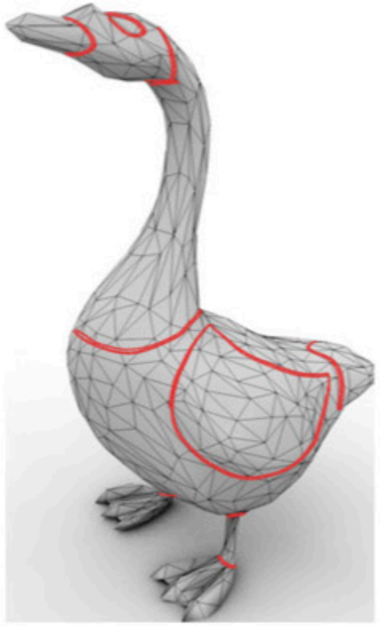


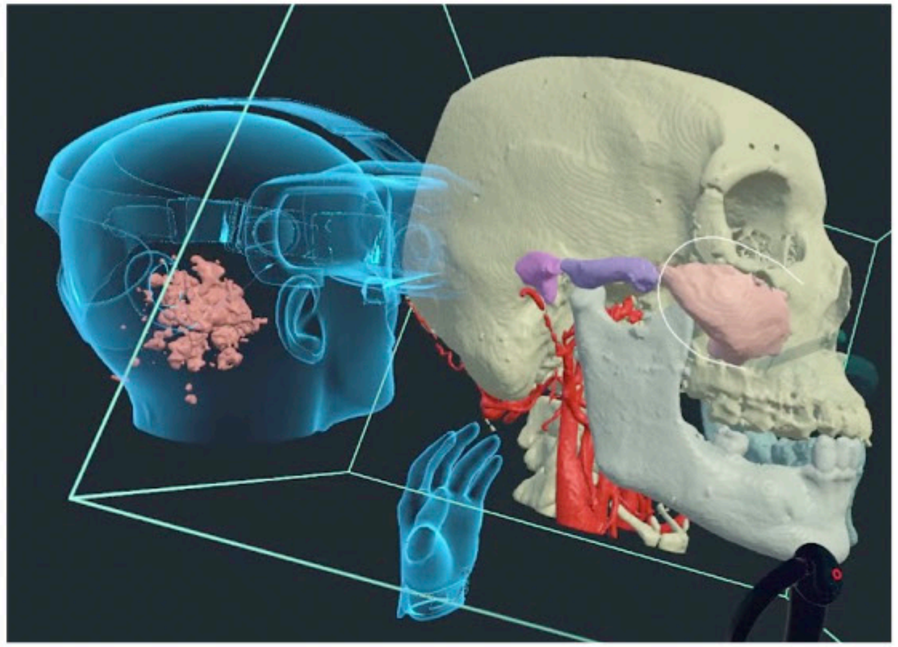


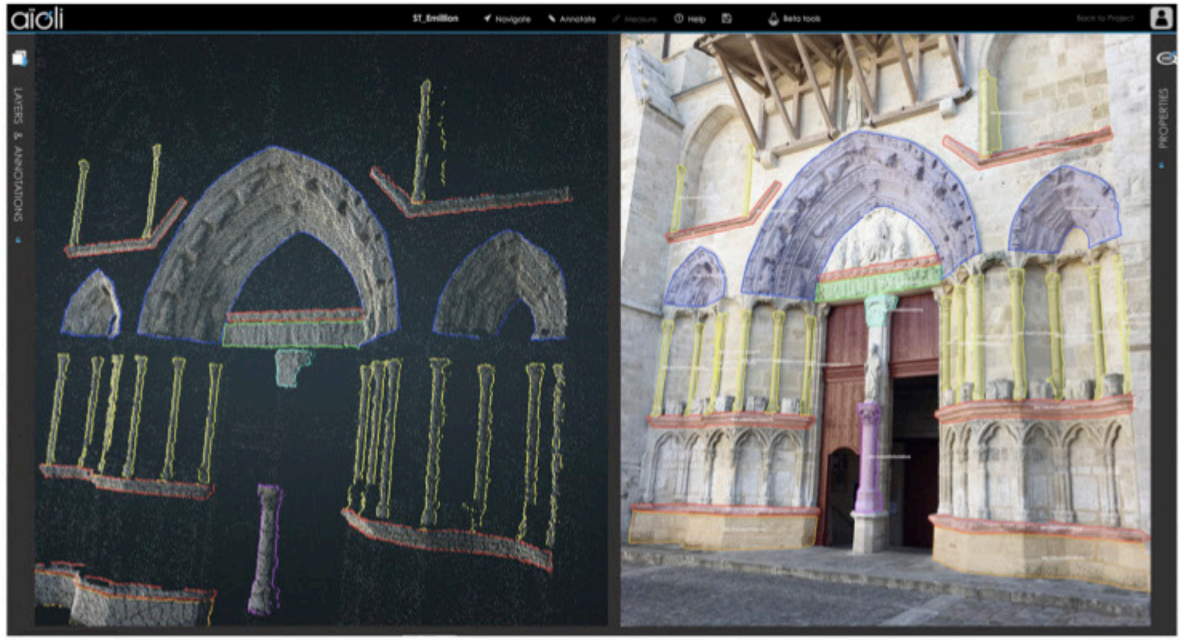


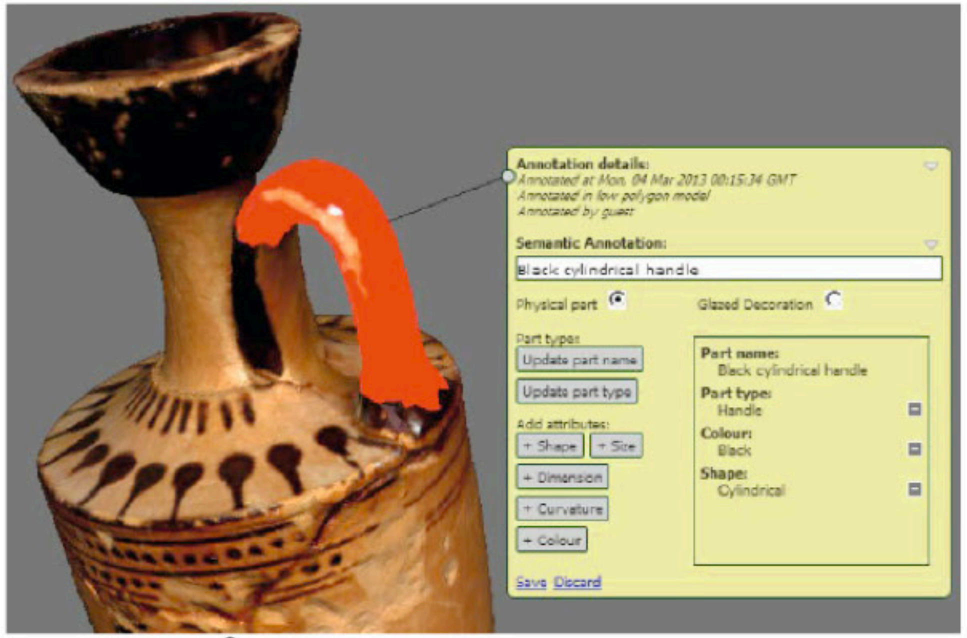


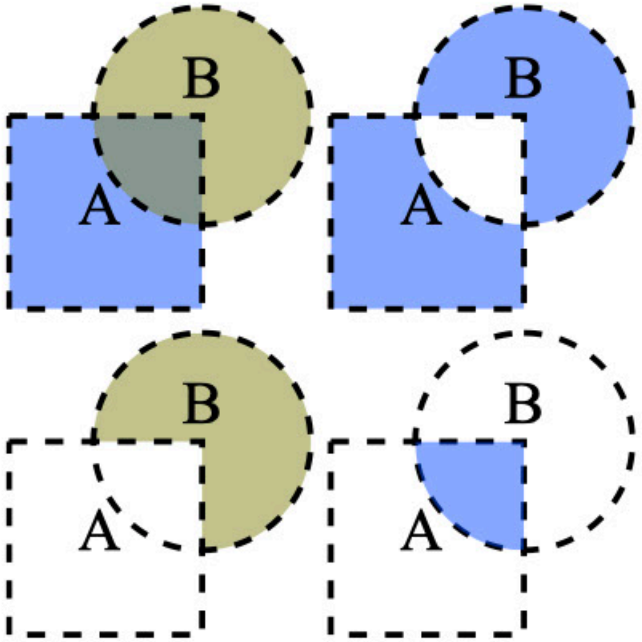
Our motivation











Past definitions: Euclidean only!

Past definitions: Euclidean only!

(Generalized) winding number = solid angle

[Euler 1781; Lagrange 1798; Gauss 1838, Maxwell 1881...]

On Solid Angles.

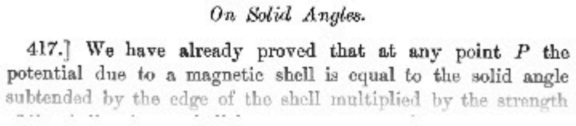
417.] We have already proved that at any point P the potential due to a magnetic shell is equal to the solid angle subtended by the edge of the shell multiplied by the strength

[Maxwell 1881]

Past definitions: Euclidean only!

(Generalized) winding number = solid angle

[Euler 1781; Lagrange 1798; Gauss 1838, Maxwell 1881…]

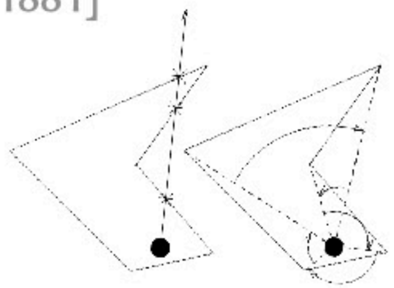

On Solid Angles.

417.] We have already proved that at any point P the potential due to a magnetic shell is equal to the solid angle subtended by the edge of the shell multiplied by the strength

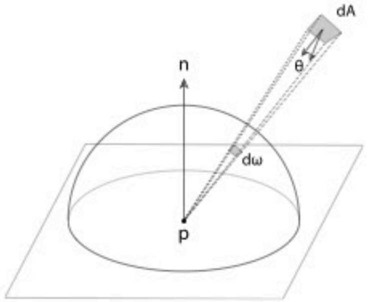
[Maxwell 1881]

Winding number & solid angle in graphics

[Shimrat 1962; Haines 1994; Goral et al. 1984; Veach & Guibas 1995…]



[Haines 1994]

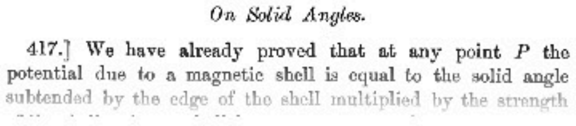


[Pharr et al. 2018]

Past definitions: Euclidean only!

(Generalized) winding number = solid angle

[Euler 1781; Lagrange 1798; Gauss 1838, Maxwell 1881…]

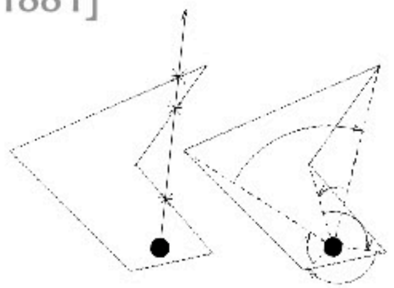

On Solid Angles.

417.] We have already proved that at any point P the potential due to a magnetic shell is equal to the solid angle subtended by the edge of the shell multiplied by the strength

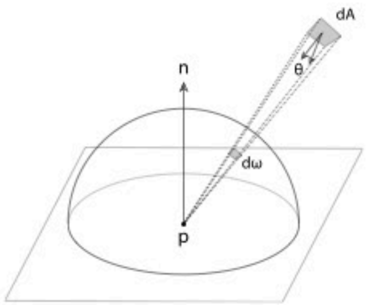
[Maxwell 1881]

Winding number & solid angle in graphics

[Shimrat 1962; Haines 1994; Goral et al. 1984; Veach & Guibas 1995…]



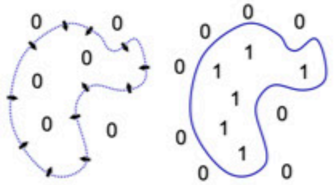
[Haines 1994]

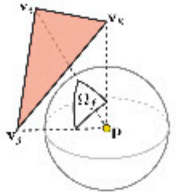


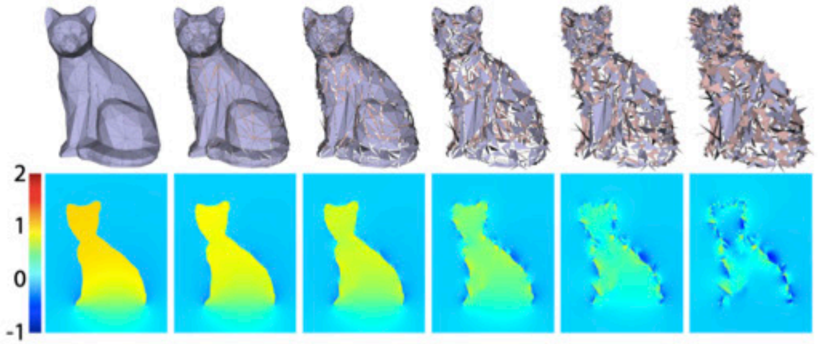
[Pharr et al. 2018]

Poisson Surface Reconstruction = Generalized Winding Number

[Kazhdan et al. 2006] [Jacobson et al. 2013]





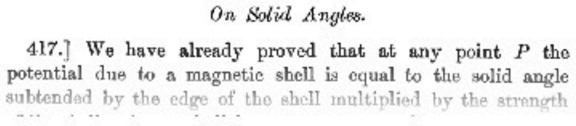


[Jacobson et al. 2013]

Past definitions: Euclidean only!

(Generalized) winding number = solid angle

[Euler 1781; Lagrange 1798; Gauss 1838, Maxwell 1881...]

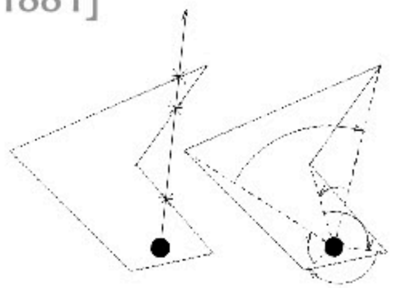

On Solid Angles.

417.] We have already proved that at any point P the potential due to a magnetic shell is equal to the solid angle subtended by the edge of the shell multiplied by the strength

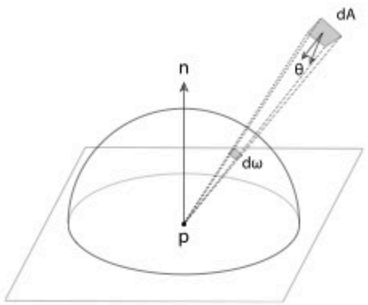
[Maxwell 1881]

Winding number & solid angle in graphics

[Shimrat 1962; Haines 1994; Goral et al. 1984; Veach & Guibas 1995...]



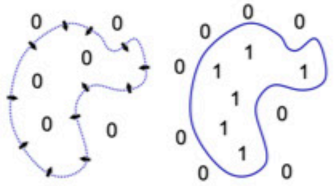
[Haines 1994]

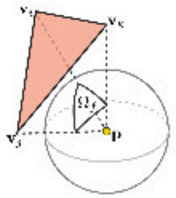


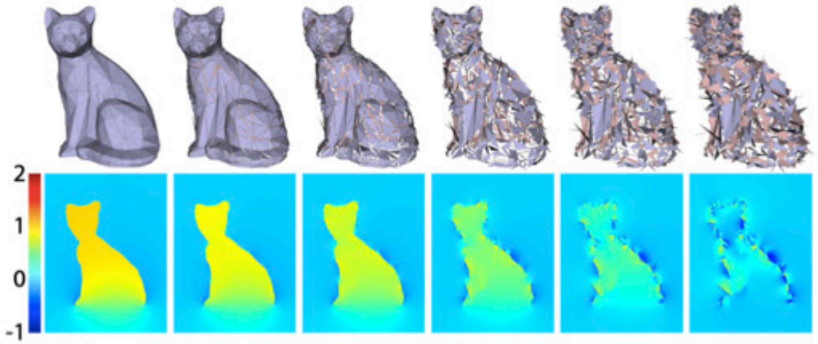
[Pharr et al. 2018]

Poisson Surface Reconstruction = Generalized Winding Number

[Kazhdan et al. 2006] [Jacobson et al. 2013]







[Jacobson et al. 2013]

~~Winding~~ Turning number on surfaces

[Reinhart 1960, 1963; Chillingworth 1972; Humphries & Johnson 1989; McIntyre & Cairns 1993; Chernov & Rudyak 2009]

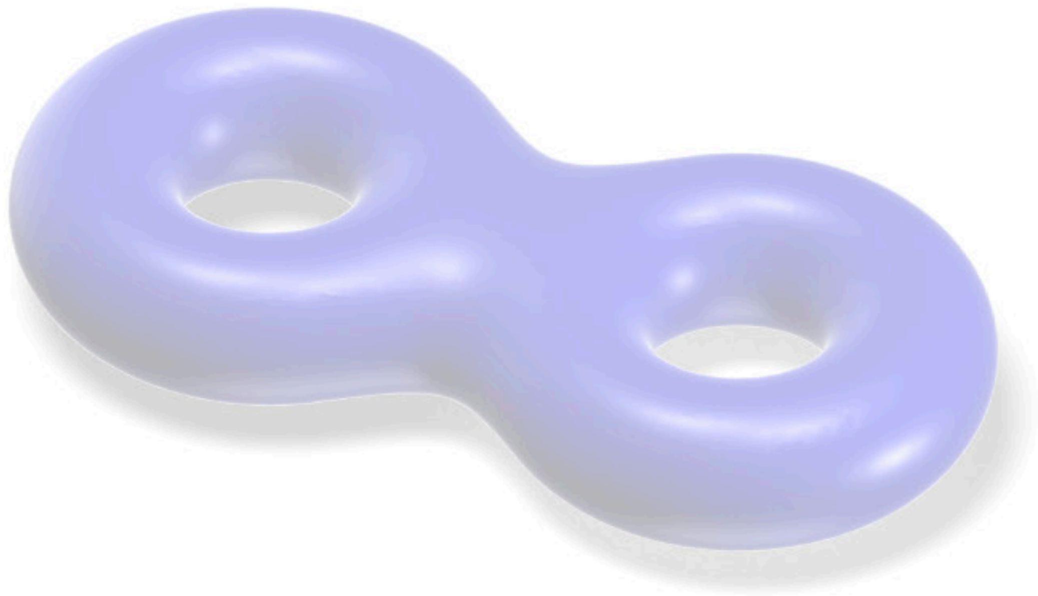
Classic methods fail

Poisson Surface Reconstruction (PSR) & *Generalized Winding Number (GWN)*
don't always identify well-defined regions.

Poisson Surface Reconstruction. Kazhdan, Bolitho, Hoppe (2006)
Robust Inside-Outside Segmentation using Generalized Winding Numbers. Jacobson, Kavan, Sorkine-Hornung (2013)

Classic methods fail

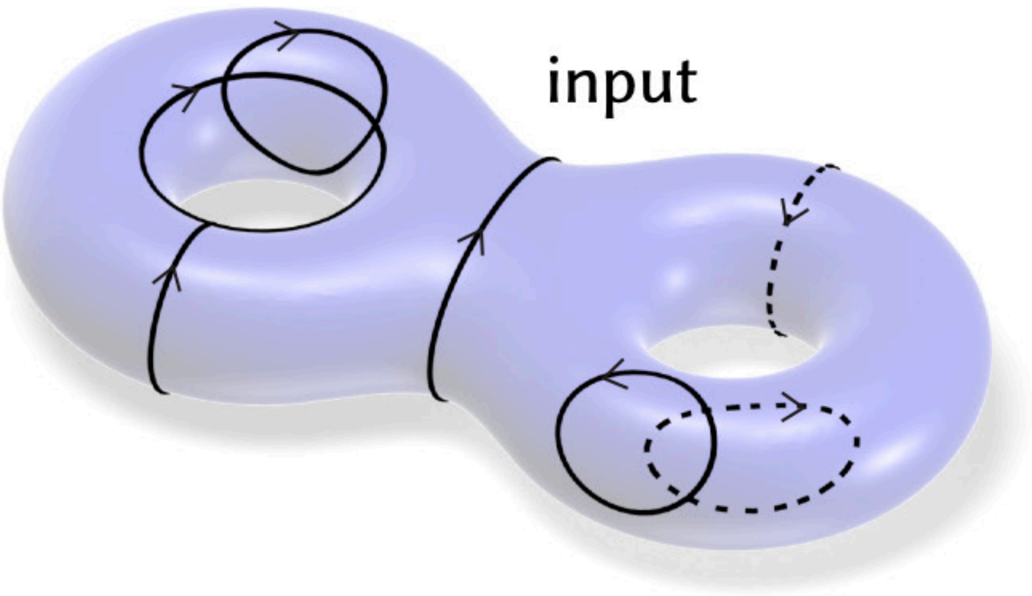
Poisson Surface Reconstruction (PSR) & Generalized Winding Number (GWN)
don't always identify well-defined regions.



Poisson Surface Reconstruction. Kazhdan, Bolitho, Hoppe (2006)
Robust Inside-Outside Segmentation using Generalized Winding Numbers. Jacobson, Kavan, Sorkine-Hornung (2013)

Classic methods fail

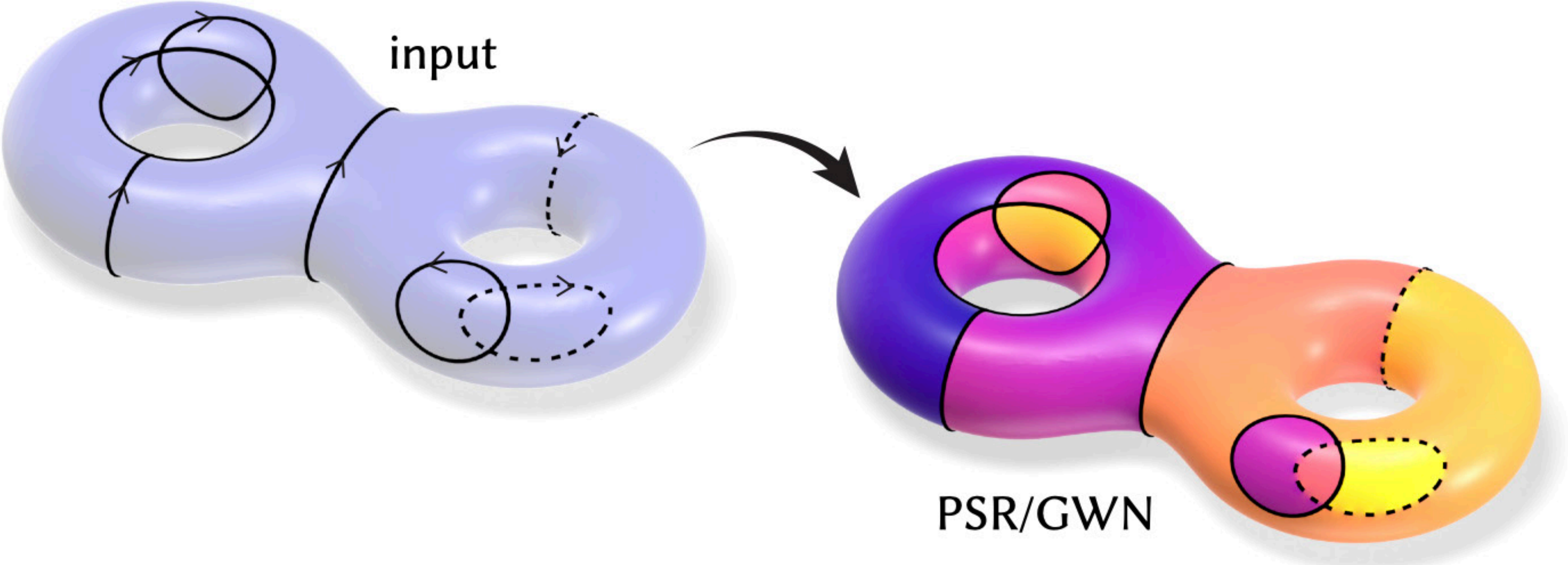
Poisson Surface Reconstruction (PSR) & *Generalized Winding Number (GWN)*
don't always identify well-defined regions.



Poisson Surface Reconstruction. Kazhdan, Bolitho, Hoppe (2006)
Robust Inside-Outside Segmentation using Generalized Winding Numbers. Jacobson, Kavan, Sorkine-Hornung (2013)

Classic methods fail

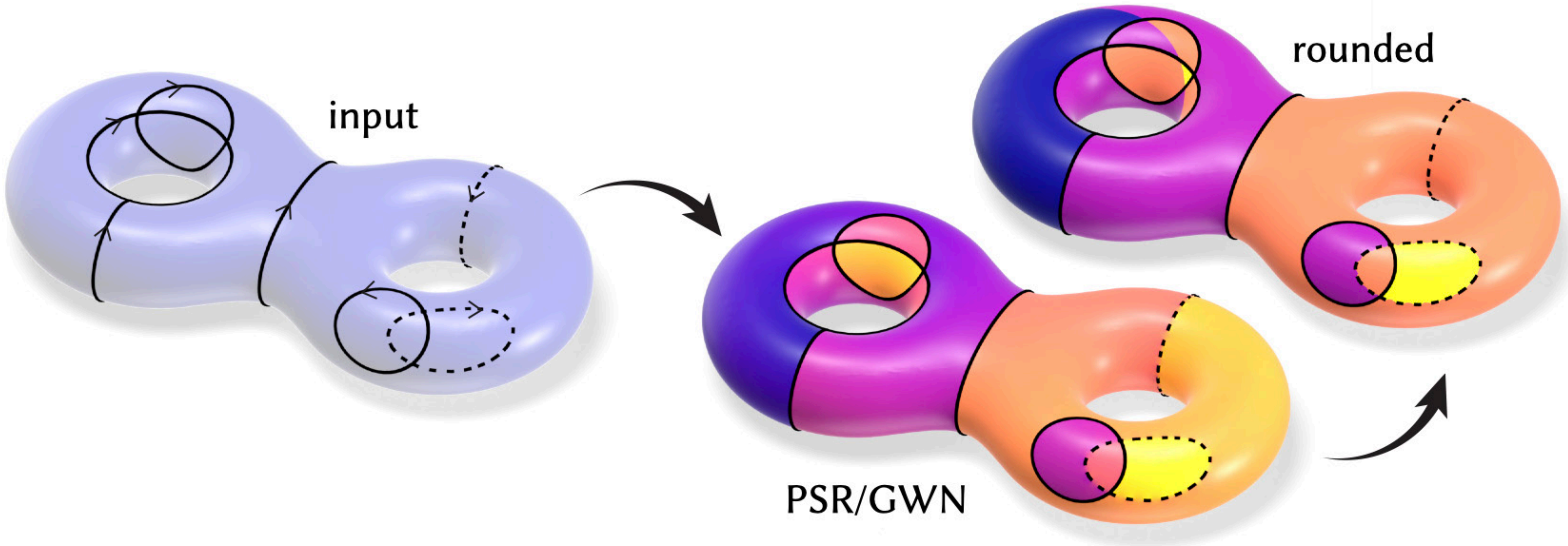
Poisson Surface Reconstruction (PSR) & Generalized Winding Number (GWN)
don't always identify well-defined regions.



Poisson Surface Reconstruction. Kazhdan, Bolitho, Hoppe (2006)
Robust Inside-Outside Segmentation using Generalized Winding Numbers. Jacobson, Kavan, Sorkine-Hornung (2013)

Classic methods fail

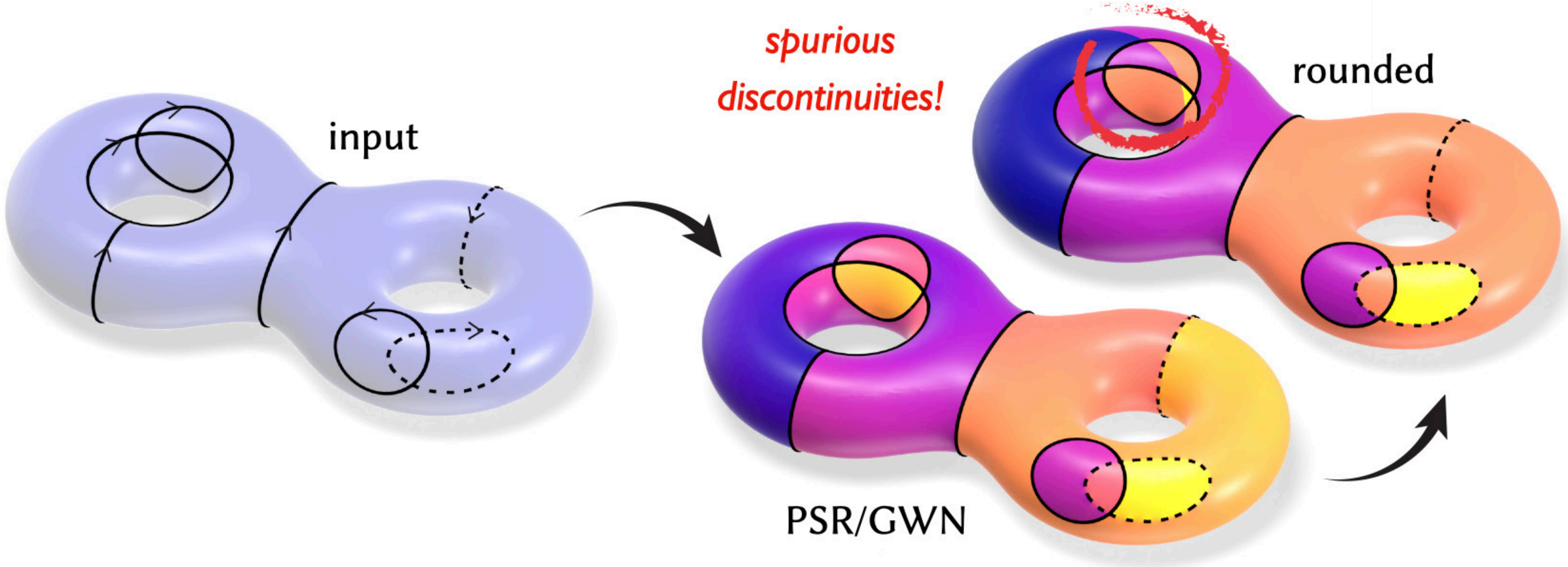
Poisson Surface Reconstruction (PSR) & Generalized Winding Number (GWN)
don't always identify well-defined regions.



Poisson Surface Reconstruction. Kazhdan, Bolitho, Hoppe (2006)
Robust Inside-Outside Segmentation using Generalized Winding Numbers. Jacobson, Kavan, Sorkine-Hornung (2013)

Classic methods fail

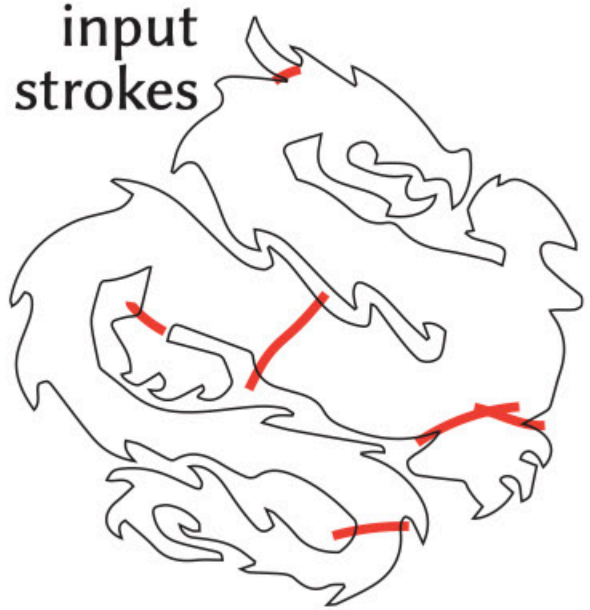
Poisson Surface Reconstruction (PSR) & Generalized Winding Number (GWN)
don't always identify well-defined regions.



Poisson Surface Reconstruction. Kazhdan, Bolitho, Hoppe (2006)
Robust Inside-Outside Segmentation using Generalized Winding Numbers. Jacobson, Kavan, Sorkine-Hornung (2013)

Classic methods fail

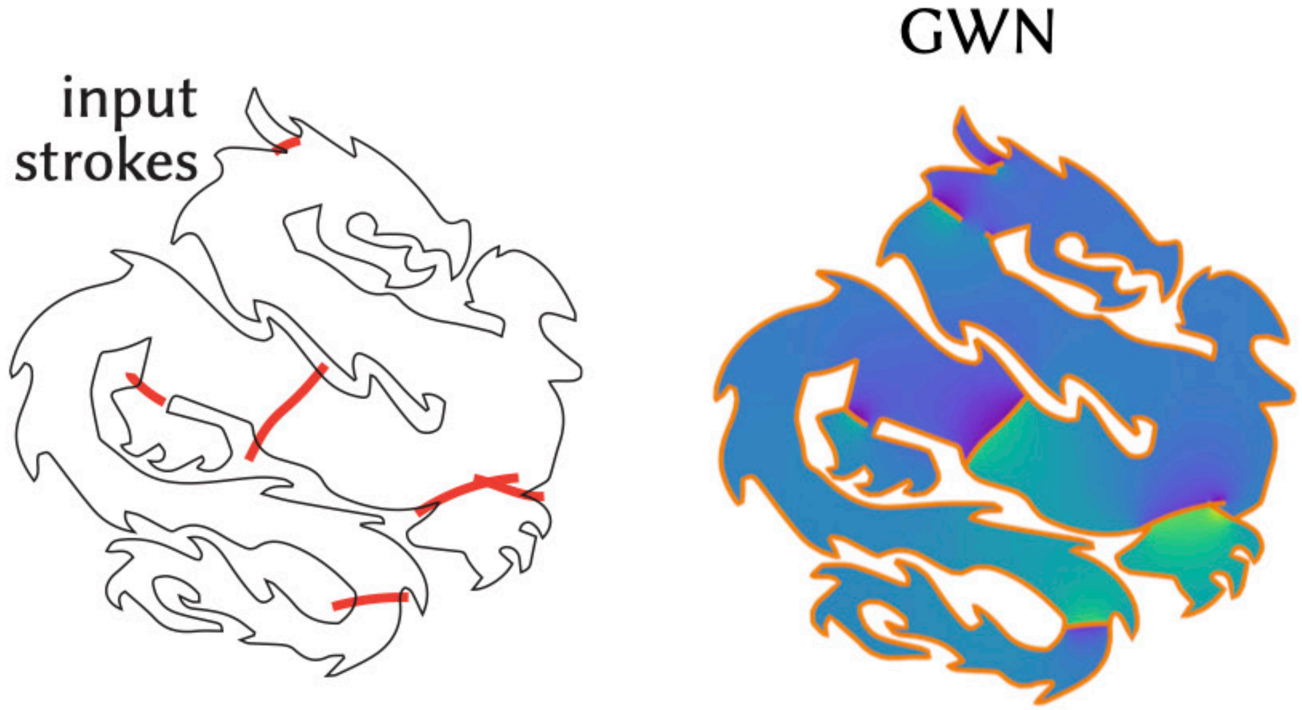
Poisson Surface Reconstruction (PSR) & Generalized Winding Number (GWN)
don't always identify well-defined regions.



Robust Inside-Outside Segmentation using Generalized Winding Numbers.
Jacobson, Kavan, Sorkine-Hornung (2013)

Classic methods fail

Poisson Surface Reconstruction (PSR) & Generalized Winding Number (GWN)
don't always identify well-defined regions.



Robust Inside-Outside Segmentation using Generalized Winding Numbers.
Jacobson, Kavan, Sorkine-Hornung (2013)

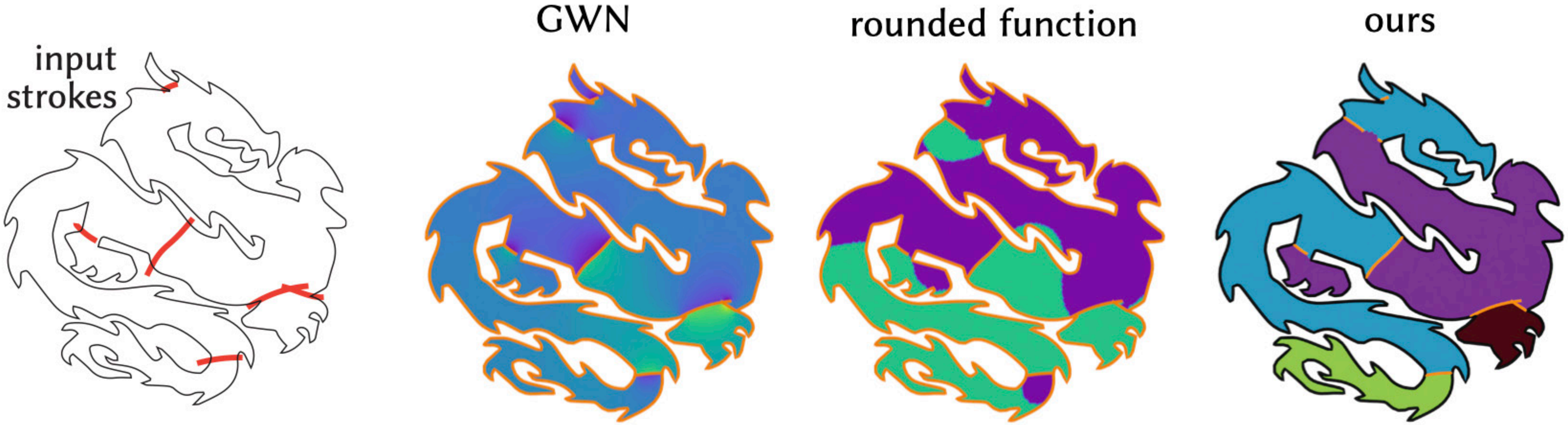
Classic methods fail

Poisson Surface Reconstruction (PSR) & Generalized Winding Number (GWN)
don't always identify well-defined regions.

Robust Inside-Outside Segmentation using Generalized Winding Numbers.
Jacobson, Kavan, Sorkine-Hornung (2013)

Classic methods fail

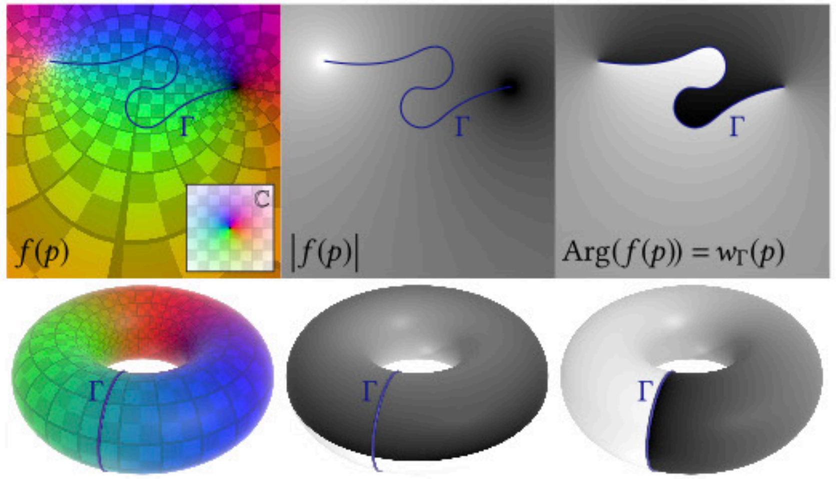
Poisson Surface Reconstruction (PSR) & Generalized Winding Number (GWN)
don't always identify well-defined regions.

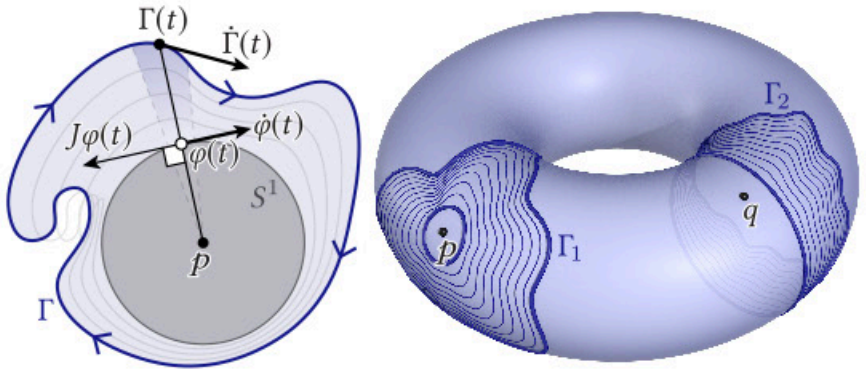


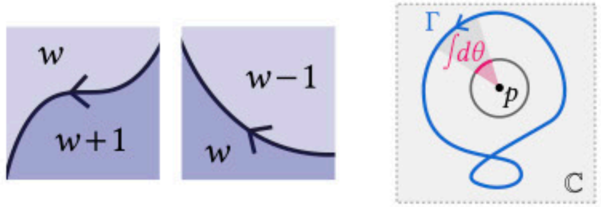
Robust Inside-Outside Segmentation using Generalized Winding Numbers.
Jacobson, Kavan, Sorkine-Hornung (2013)

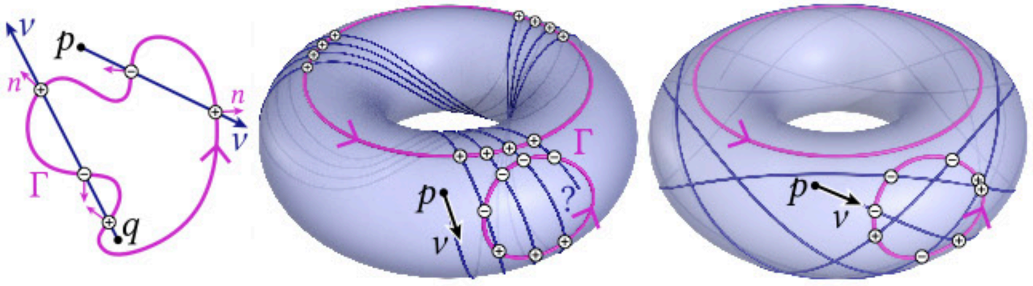
Many formulas for solid angle…

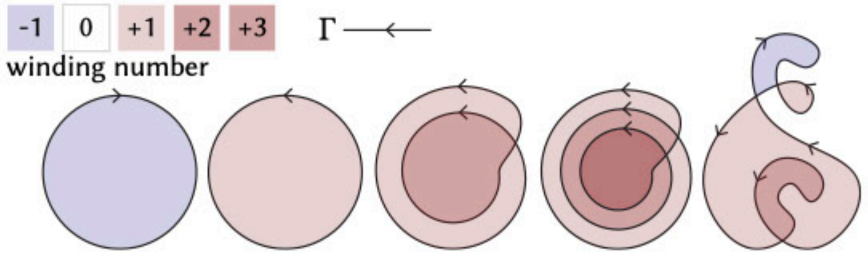
Many formulas for solid angle...

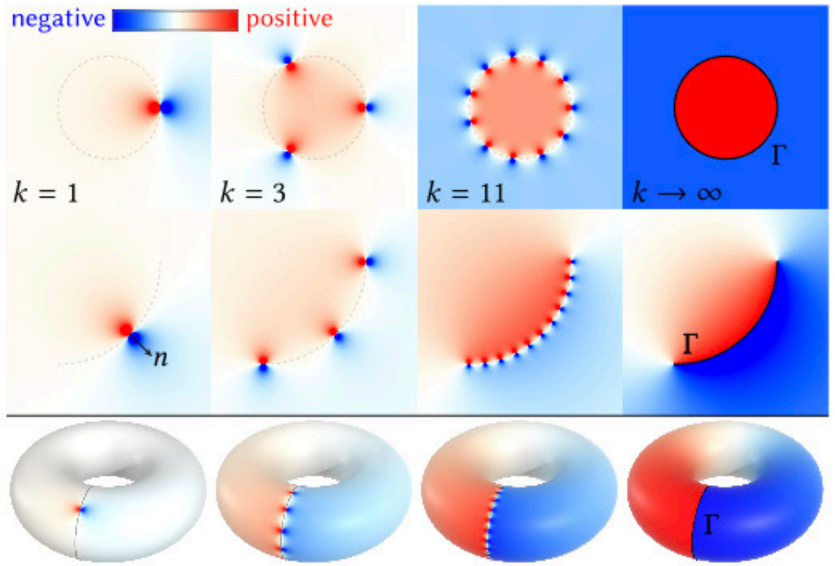


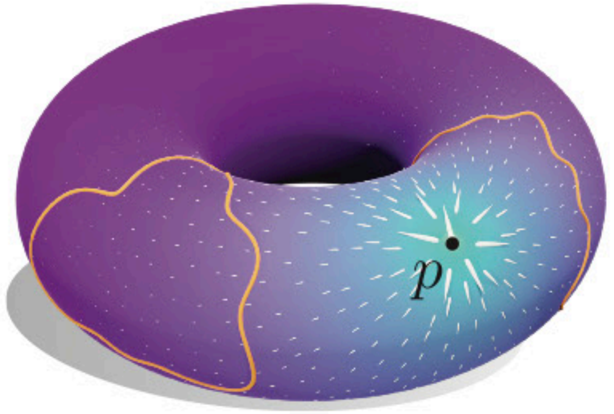


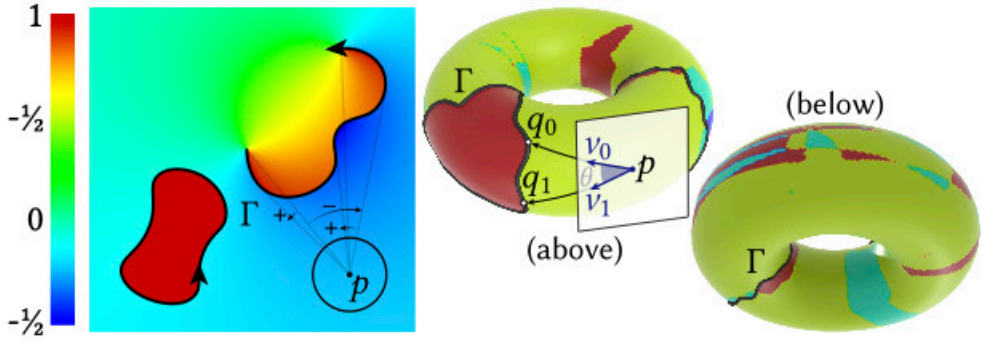


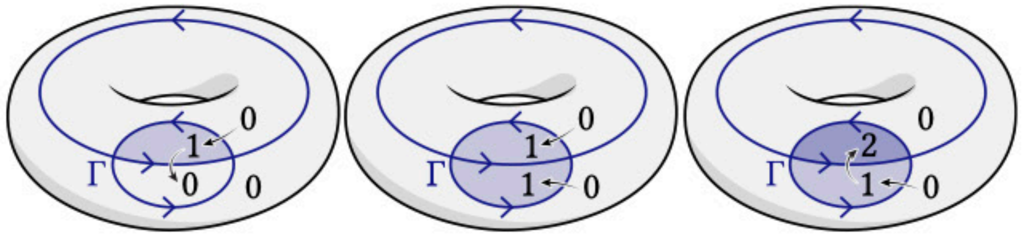




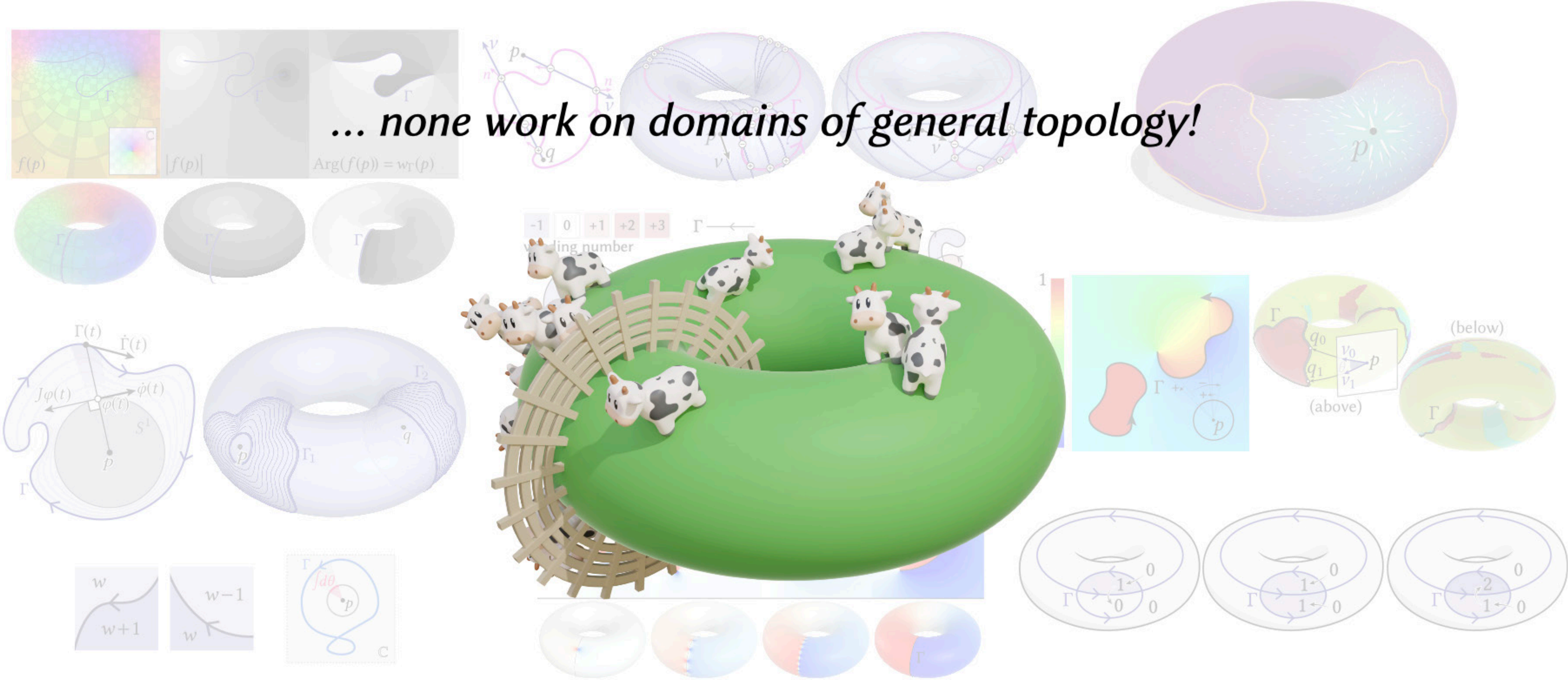








Many formulas for solid angle…



… none work on domains of general topology!

The general case is difficult!

The general case is difficult!

BoolSurf [Riso et al. 2022]:

[Riso et al. 2022]

The general case is difficult!

BoolSurf [Riso et al. 2022]:


[Riso et al. 2022]

input is already segmented into loops

The general case is difficult!

BoolSurf [Riso et al. 2022]:

[Riso et al. 2022]

input is already segmented into loops

closed loops only

The general case is difficult!

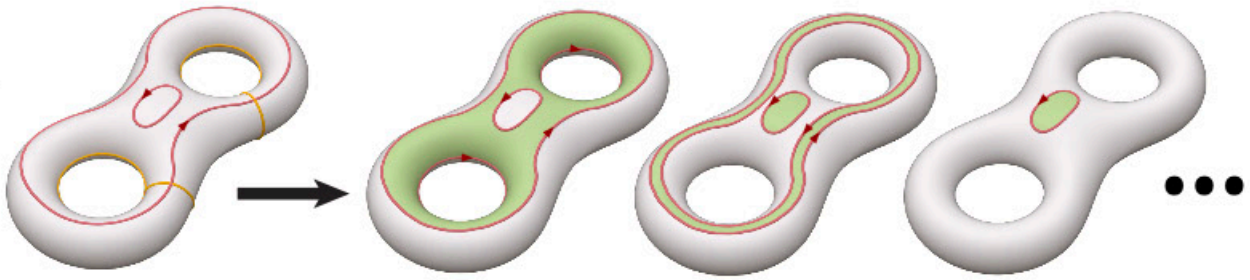
BoolSurf [Riso et al. 2022]:

[Riso et al. 2022]

input is already segmented into loops

closed loops only

requires user guidance to handle nonbounding curves



The general case is difficult!

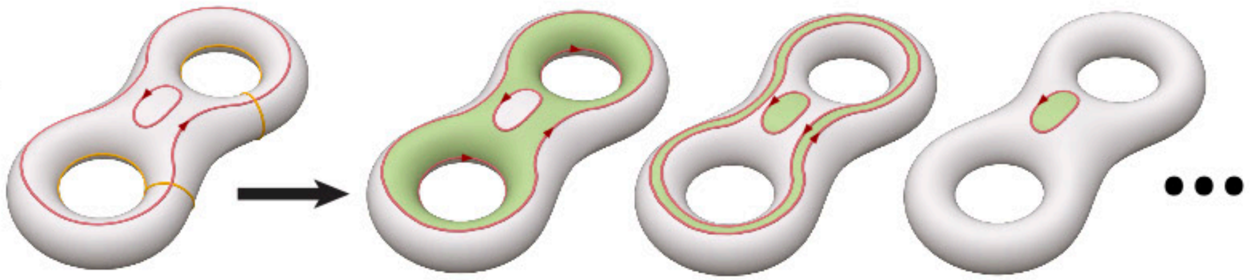
BoolSurf [Riso et al. 2022]:

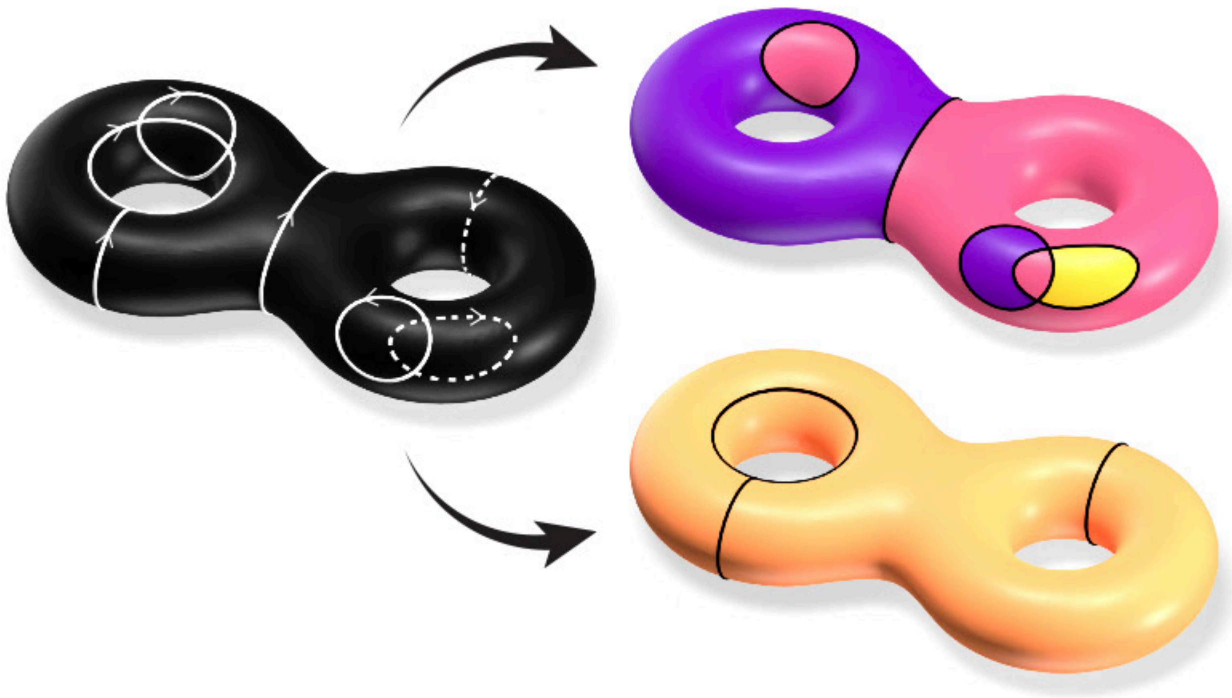
[Riso et al. 2022]

input is already segmented into loops

closed loops only

requires user guidance to handle nonbounding curves





The general case is difficult!

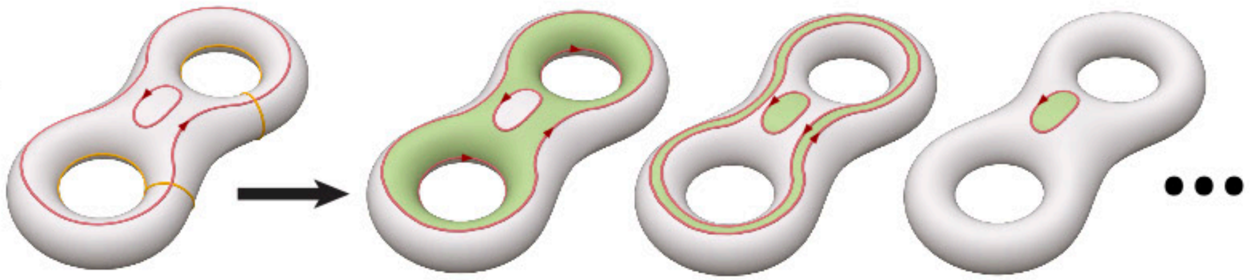
BoolSurf [Riso et al. 2022]:

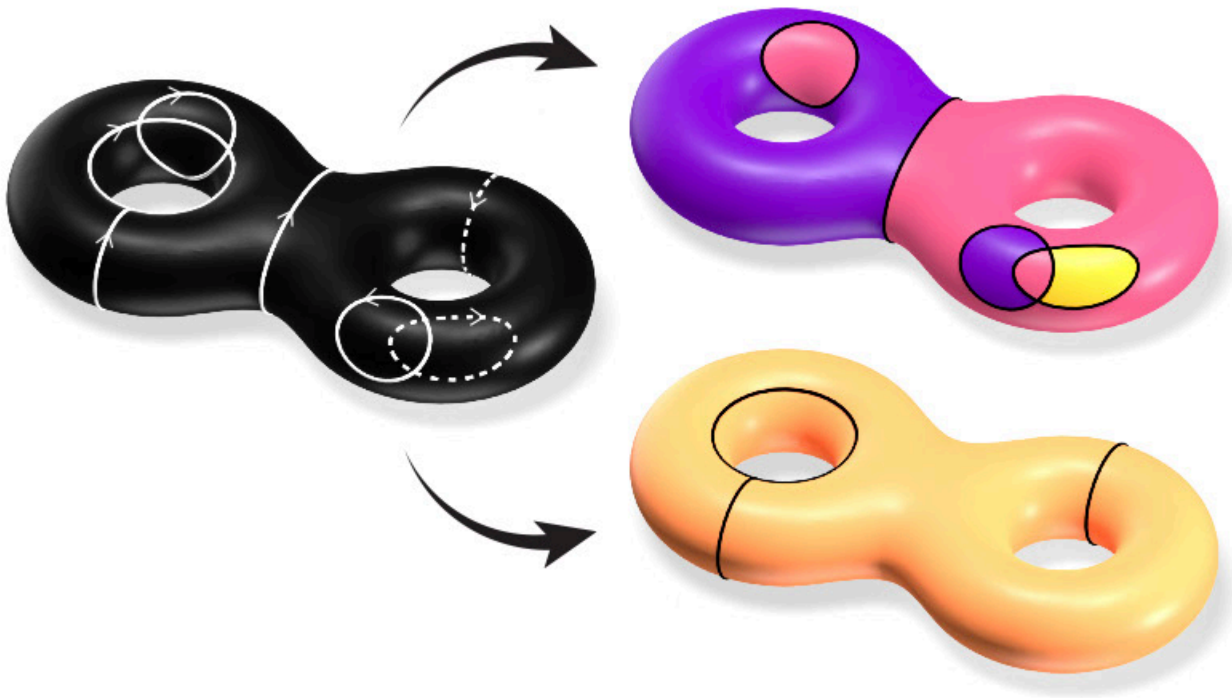
[Riso et al. 2022]

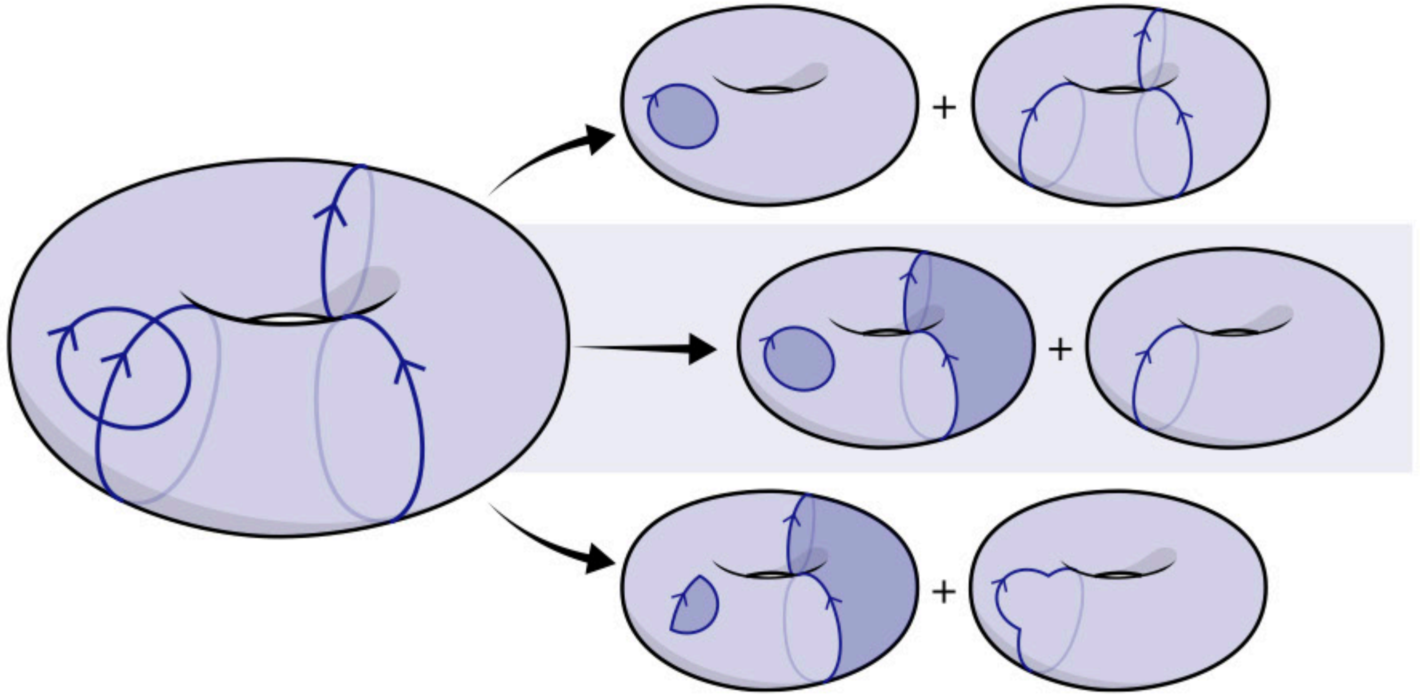
input is already segmented into loops

closed loops only

requires user guidance to handle nonbounding curves







The general case is difficult!

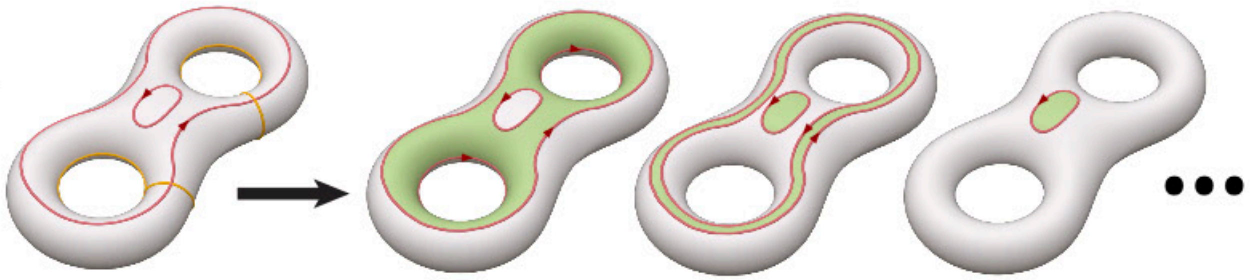
BoolSurf [Riso et al. 2022]:

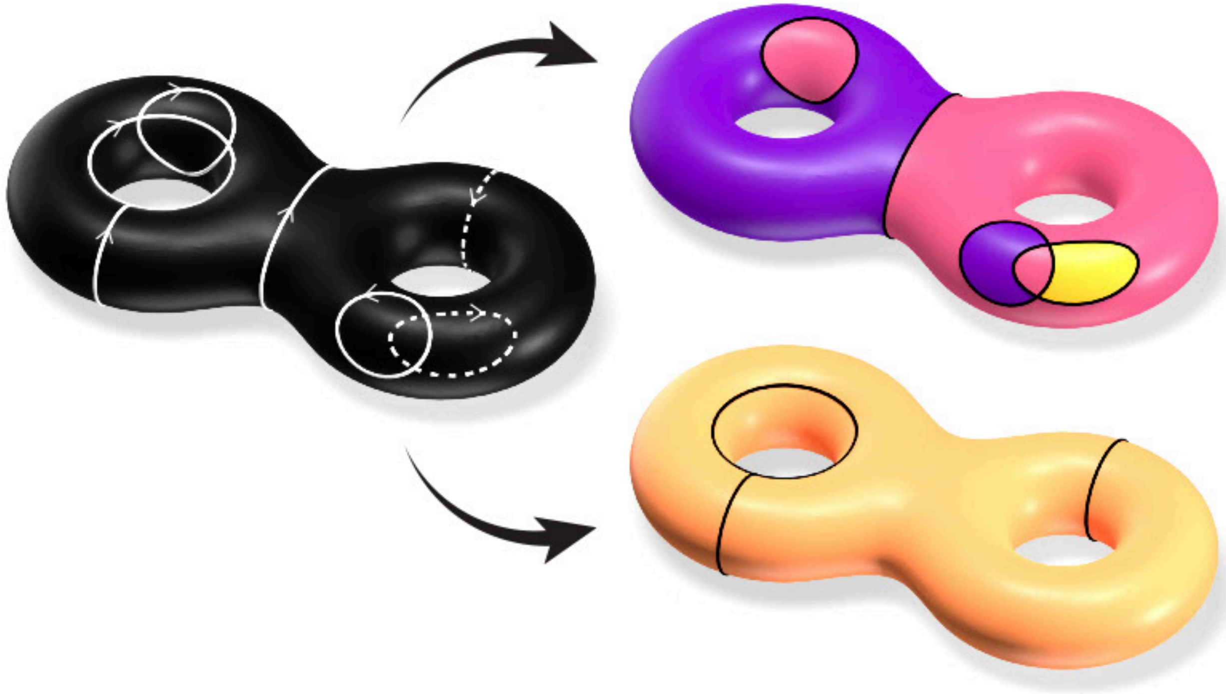
[Riso et al. 2022]

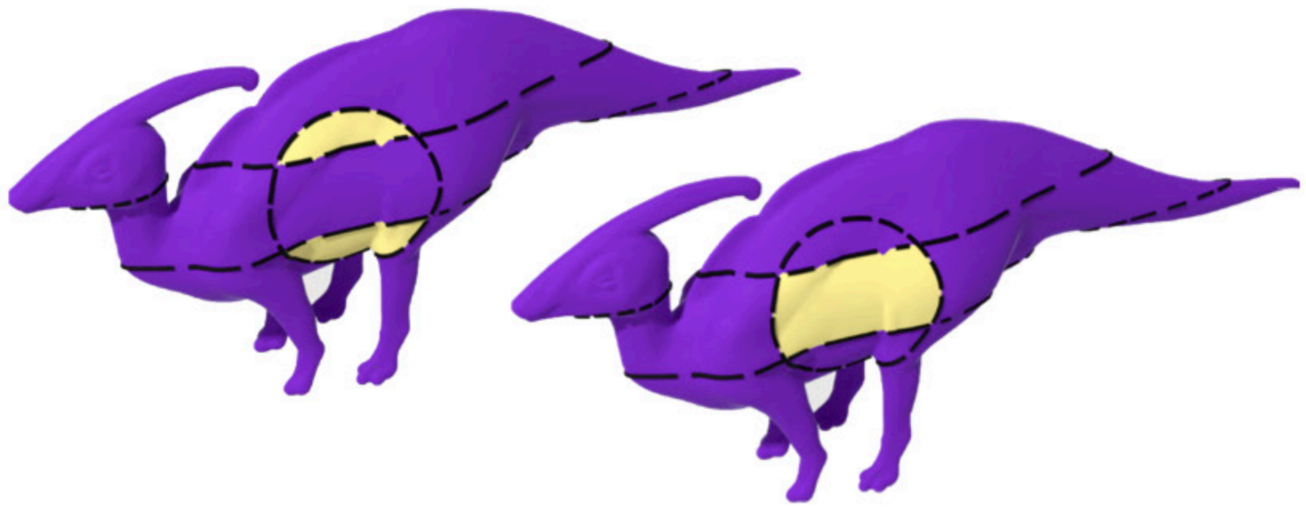
input is already segmented into loops

closed loops only

requires user guidance to handle nonbounding curves







Past approaches to "homological geometry processing"

Past approaches to "homological geometry processing"

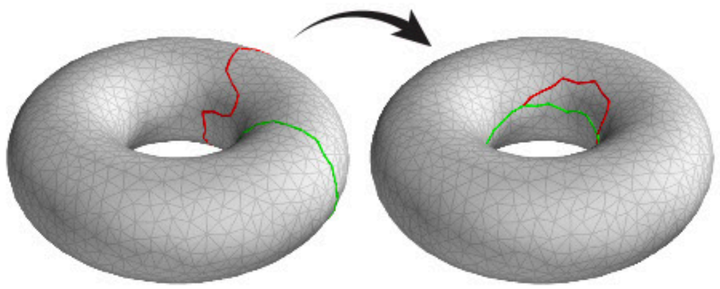
Geometry processing with homological constraints

[Born et al. 2021; Dey et al. 2010; Wang & Chern 2021]

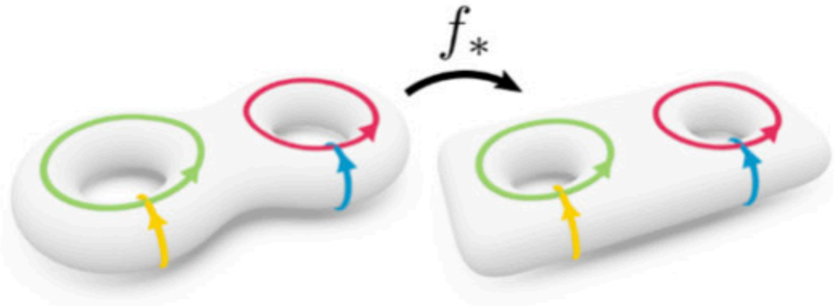
Past approaches to "homological geometry processing"

Geometry processing with homological constraints

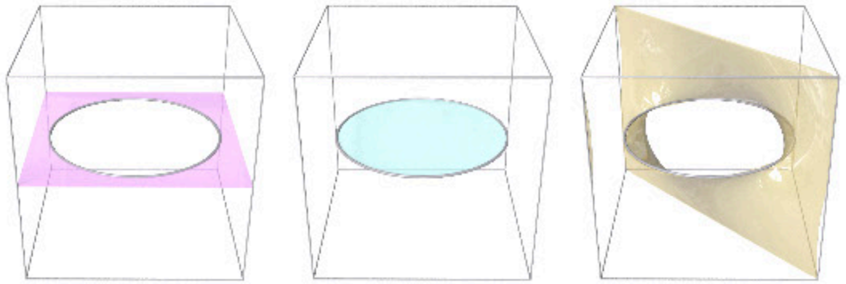
[Born et al. 2021; Dey et al. 2010; Wang & Chern 2021]



[Dey et al. 2010]



[Born et al. 2021]

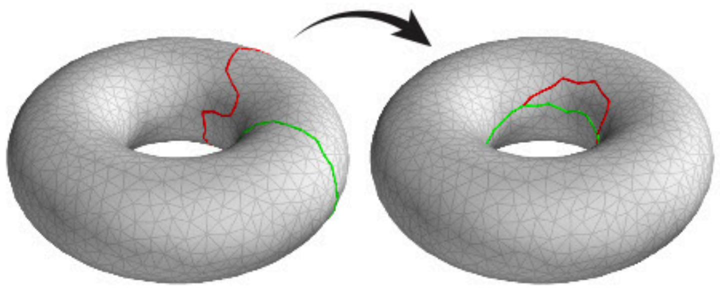


[Wang & Chern 2021]

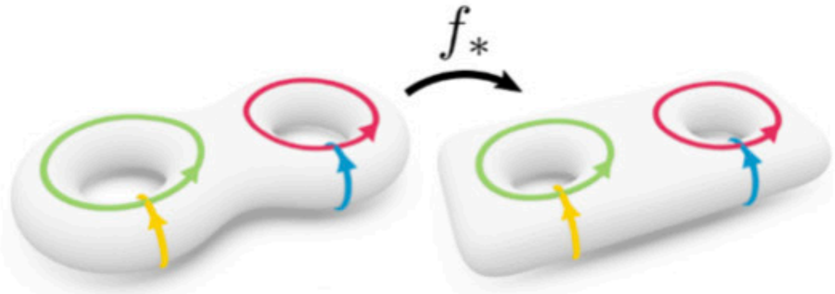
Past approaches to "homological geometry processing"

Geometry processing with homological constraints

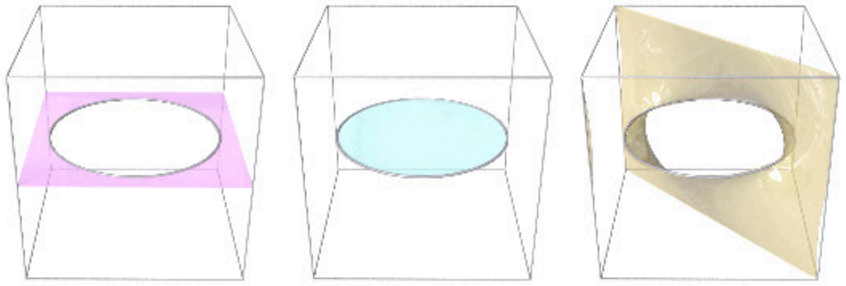
[Born et al. 2021; Dey et al. 2010; Wang & Chern 2021]



[Dey et al. 2010]

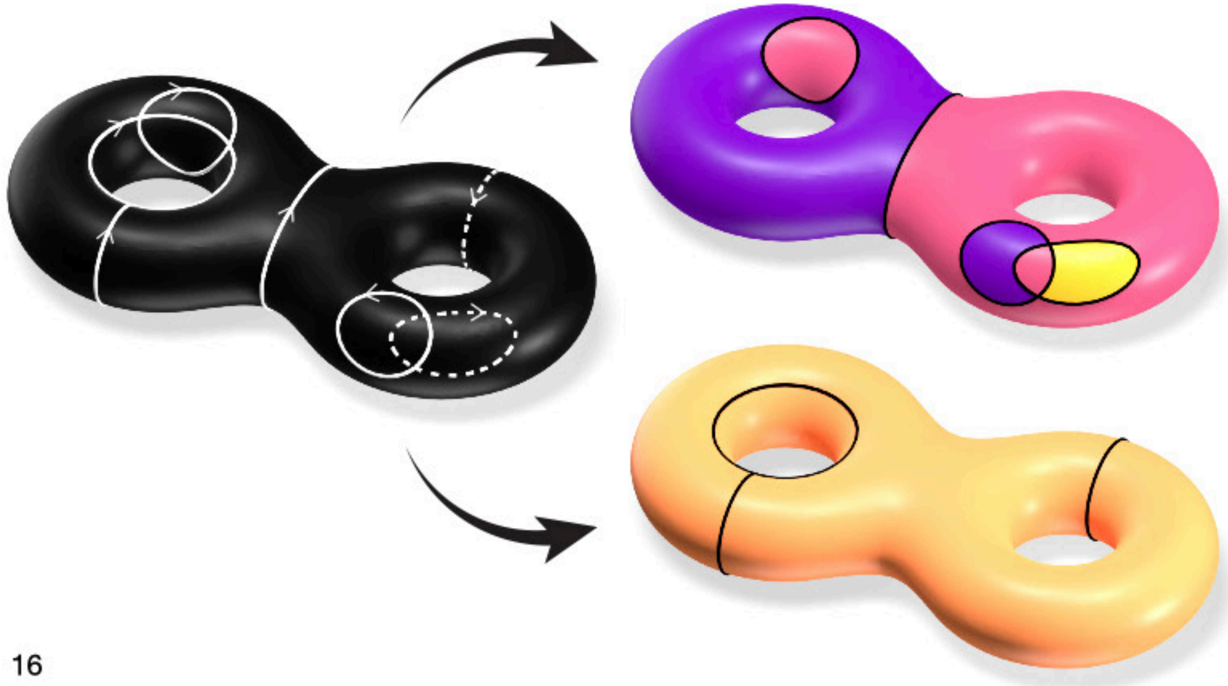


[Born et al. 2021]



[Wang & Chern 2021]

We want to *infer* curve topology!

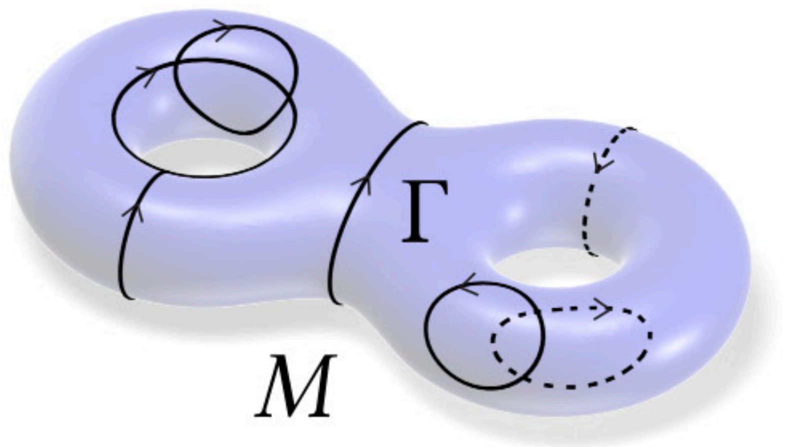


Algorithm input & output

Algorithm input & output

Input:

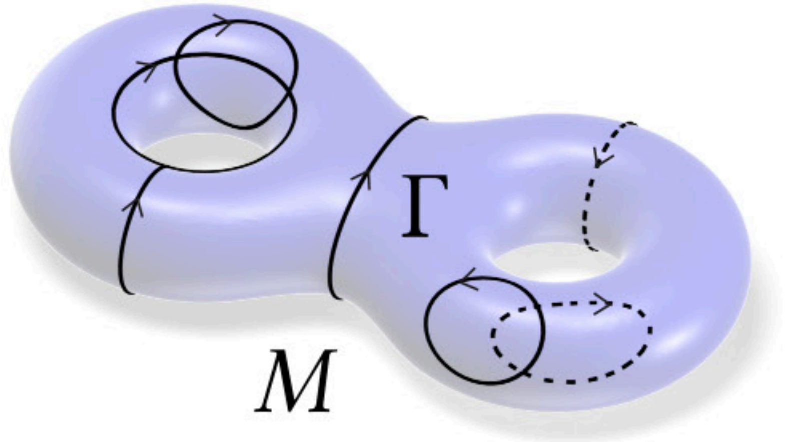
(Possibly broken) oriented curve Γ on a surface M.



Algorithm input & output

Input:

(Possibly broken) oriented curve Γ on a surface M.

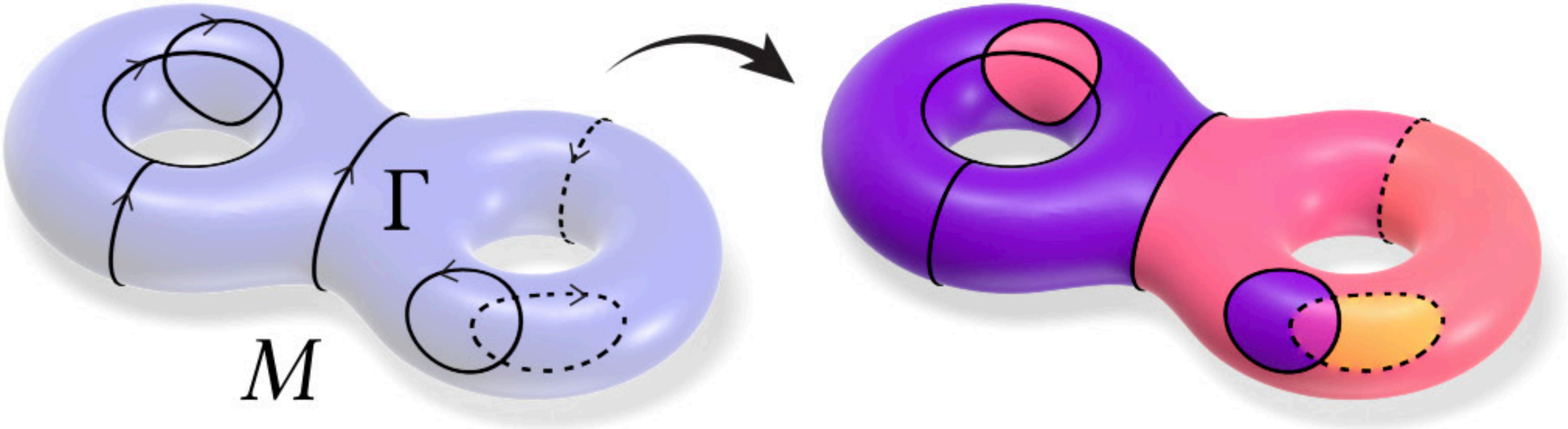


Output:

Algorithm input & output

Input:

(Possibly broken) oriented curve Γ on a surface M.



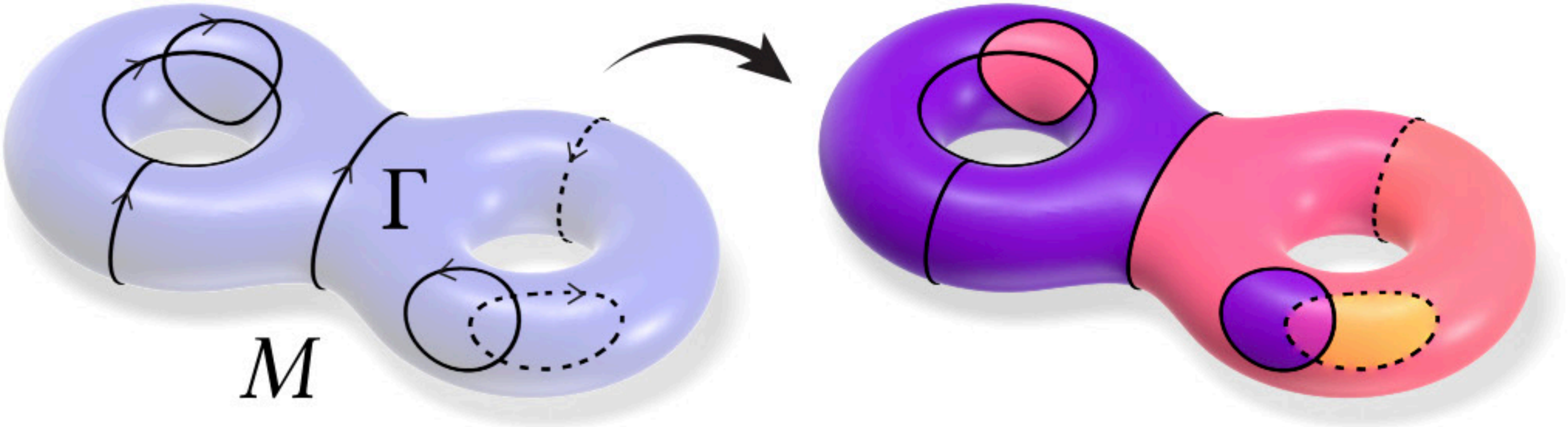
Output:

Region labels induced by bounding components of Γ.

Algorithm input & output

Input:

(Possibly broken) oriented curve Γ on a surface M.



Output:

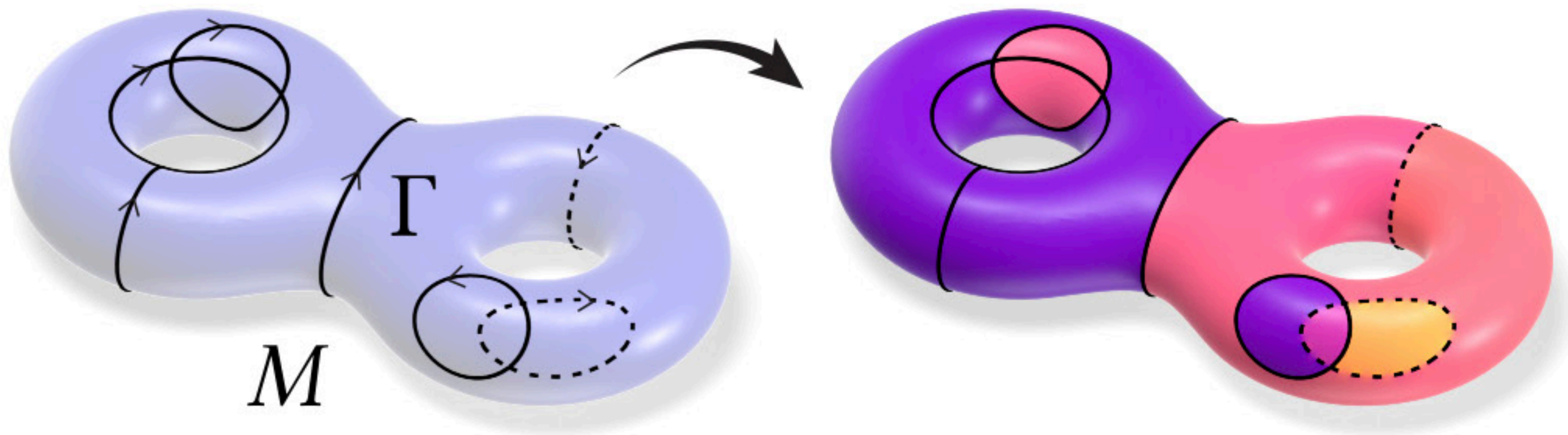
Region labels induced by bounding components of Γ.

A decomposition of Γ into:

Algorithm input & output

Input:

(Possibly broken) oriented curve Γ on a surface M.

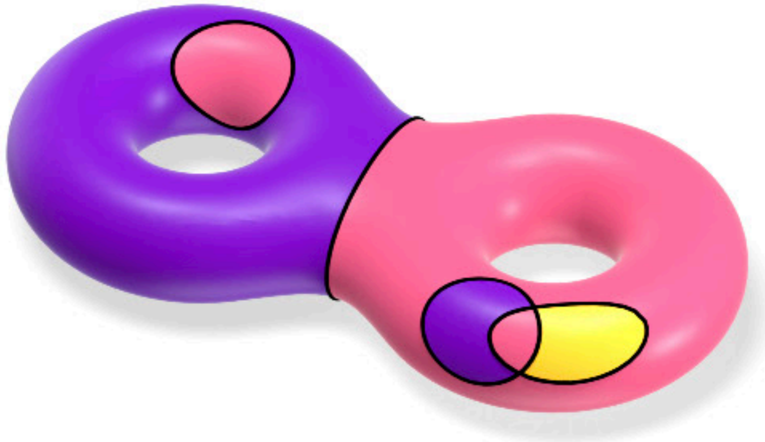


Output:

Region labels induced by bounding components of Γ.

A decomposition of Γ into:

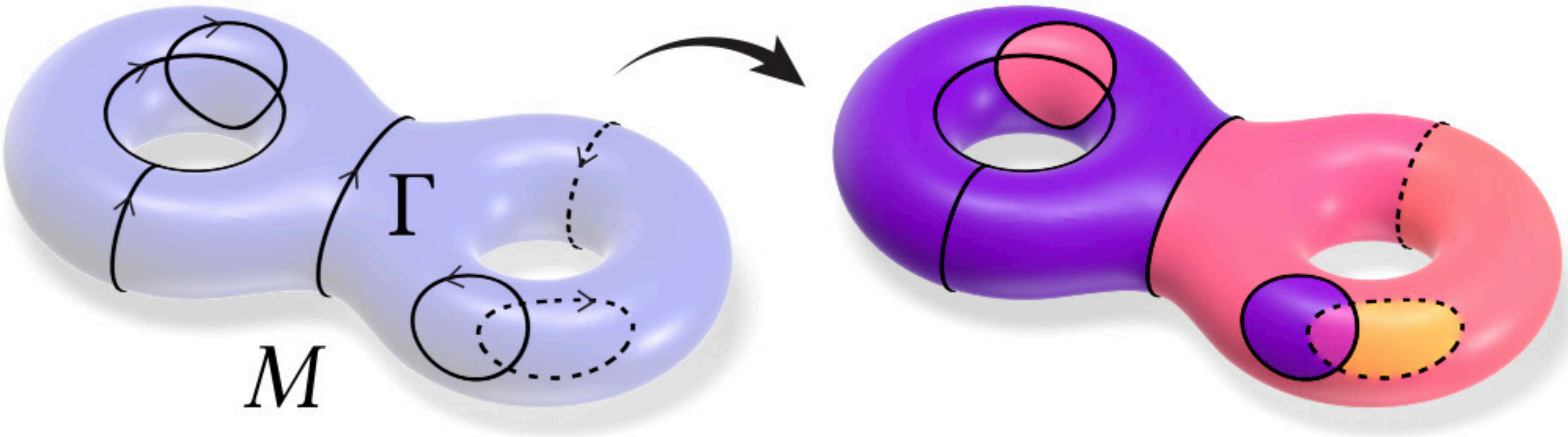
- bounding components that induce valid regions



Algorithm input & output

Input:

(Possibly broken) oriented curve Γ on a surface M.

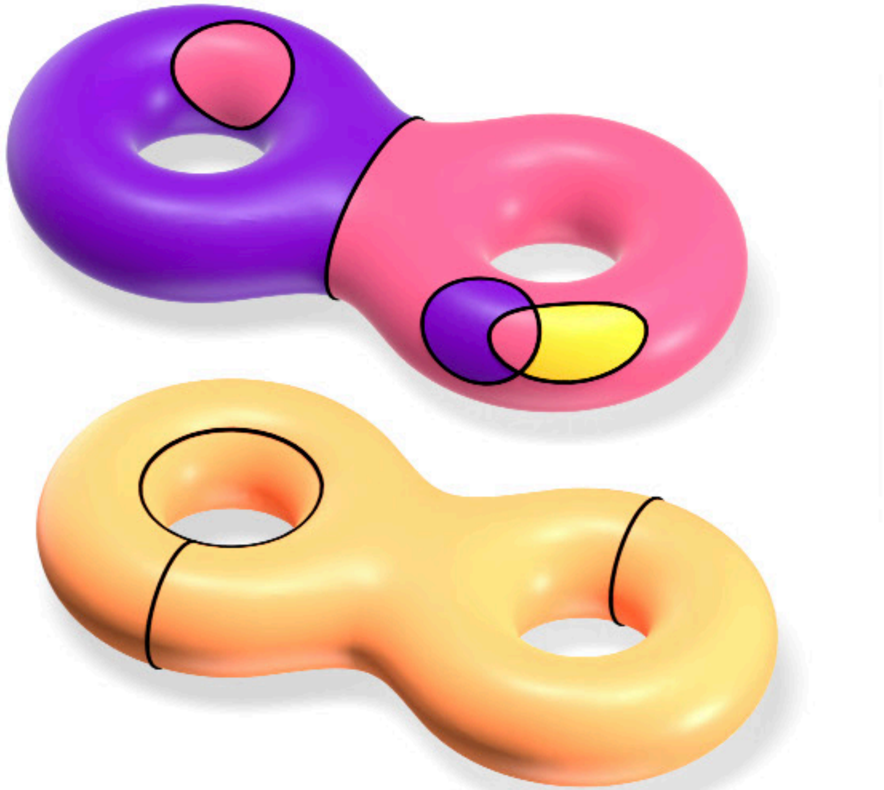


Output:

Region labels induced by bounding components of Γ.

A decomposition of Γ into:

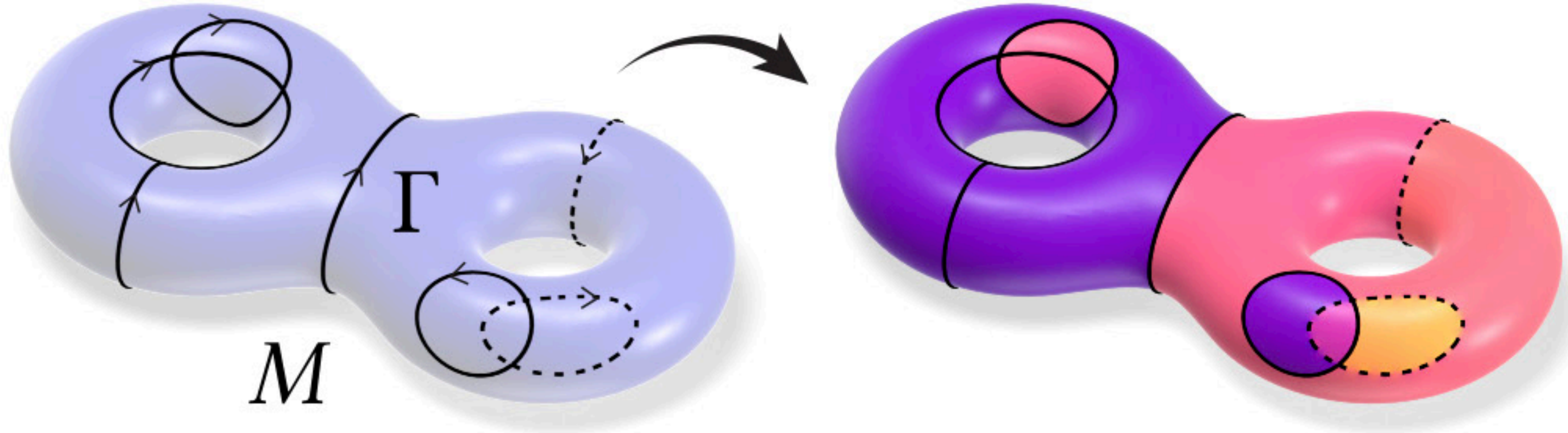
- bounding components that induce valid regions
- nonbounding components.



Algorithm input & output

Input:

(Possibly broken) oriented curve Γ on a surface M.



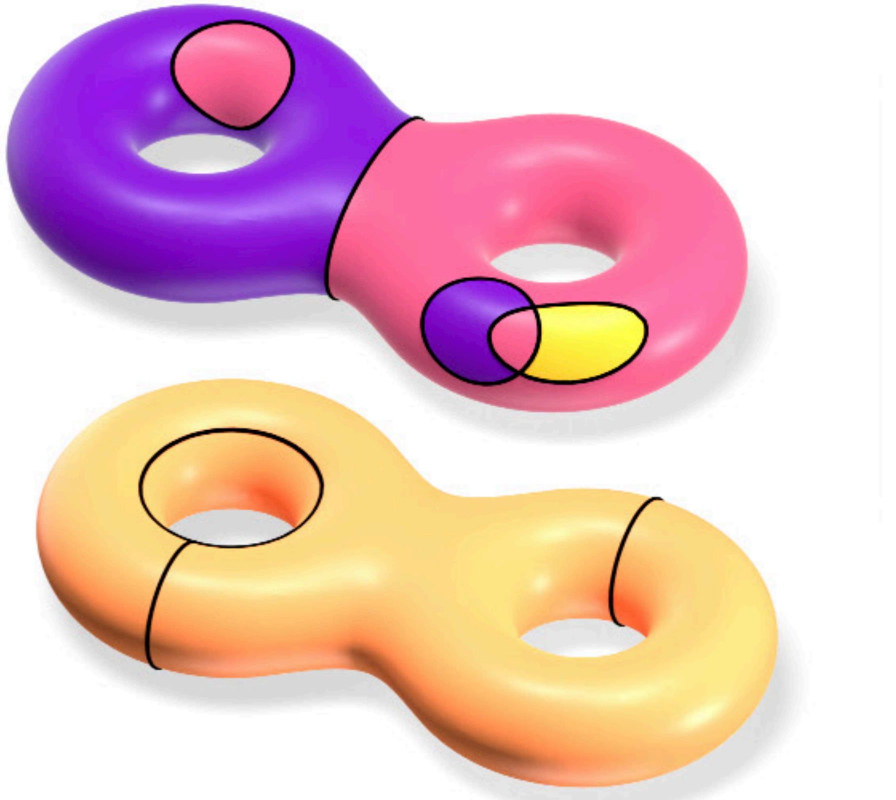
Output:

Region labels induced by bounding components of Γ.

A decomposition of Γ into:

- bounding components that induce valid regions
- nonbounding components.

A closed, completed version of the input curve.

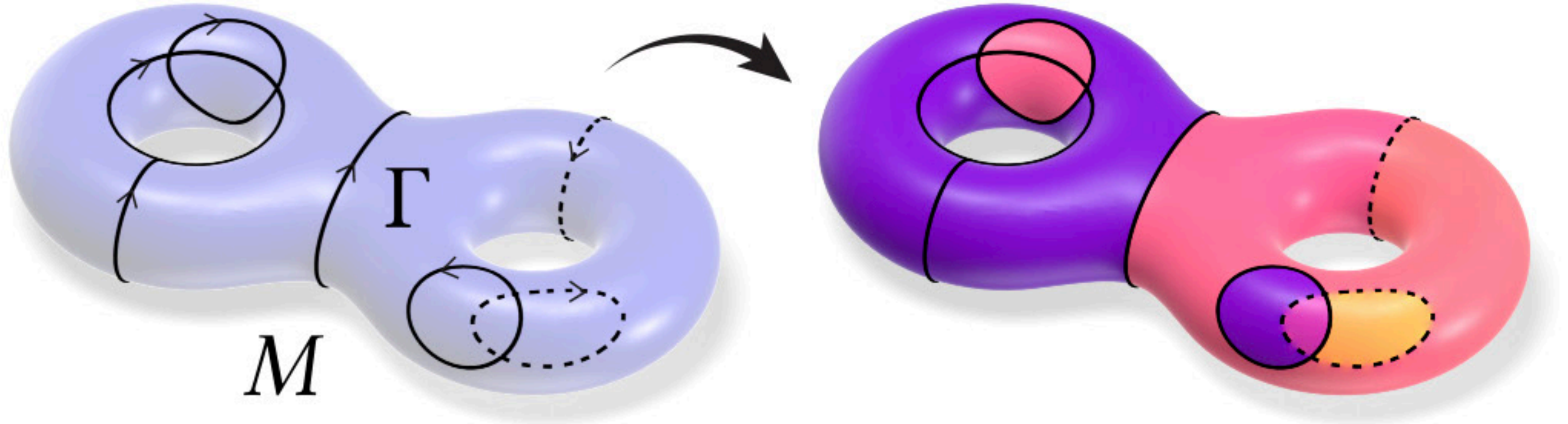


Algorithm input & output

Input:

(Possibly broken) oriented curve Γ on a surface M.

Γ is ***not*** partitioned into loops — no labels!



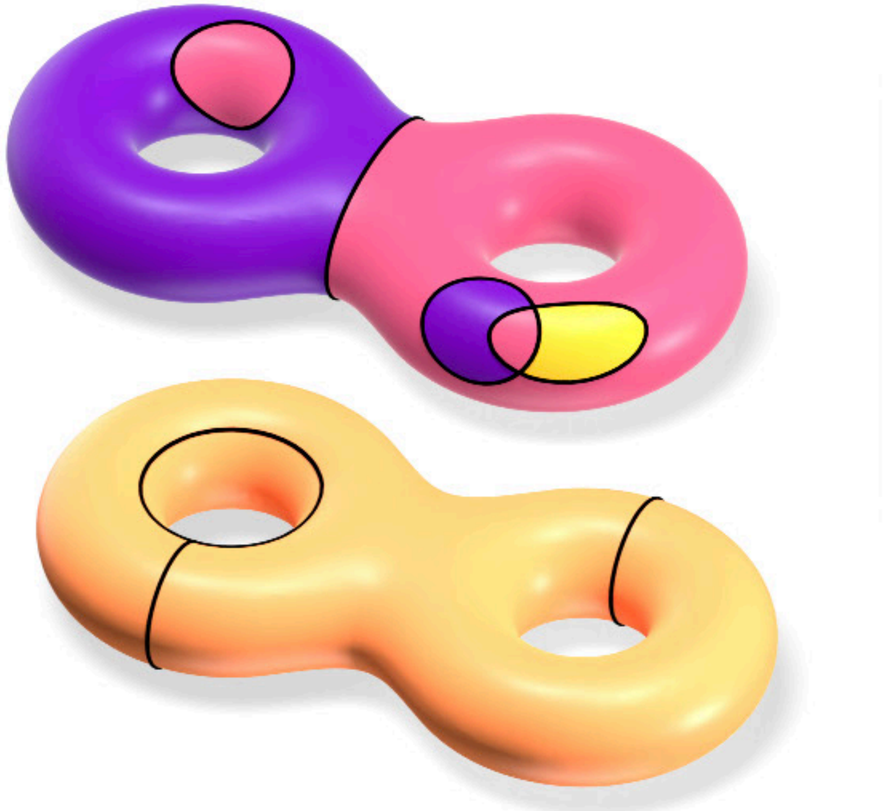
Output:

Region labels induced by bounding components of Γ.

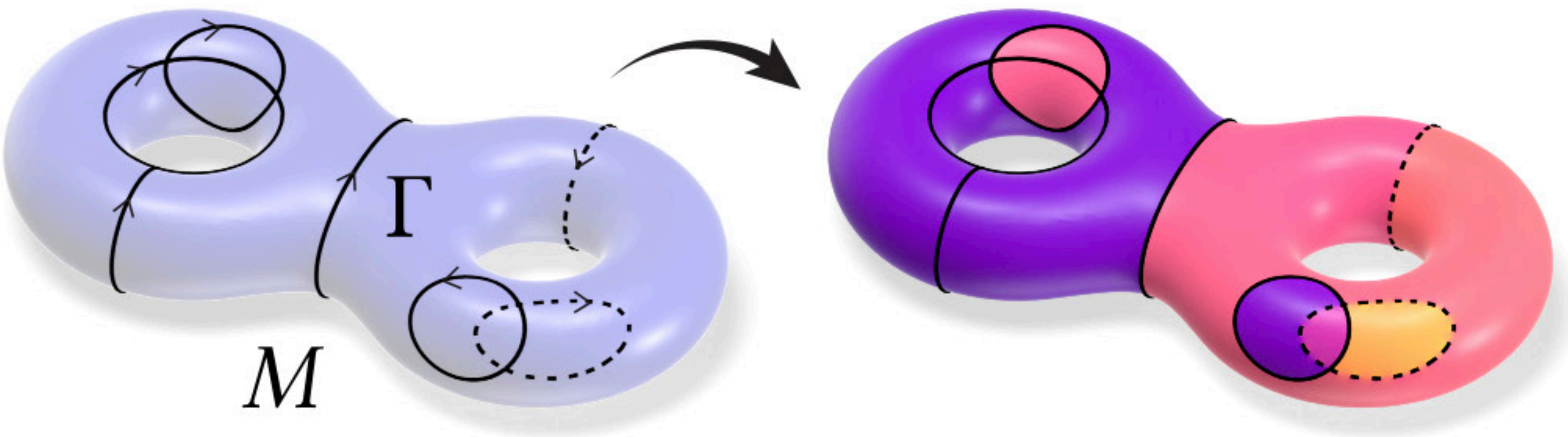
A decomposition of Γ into:

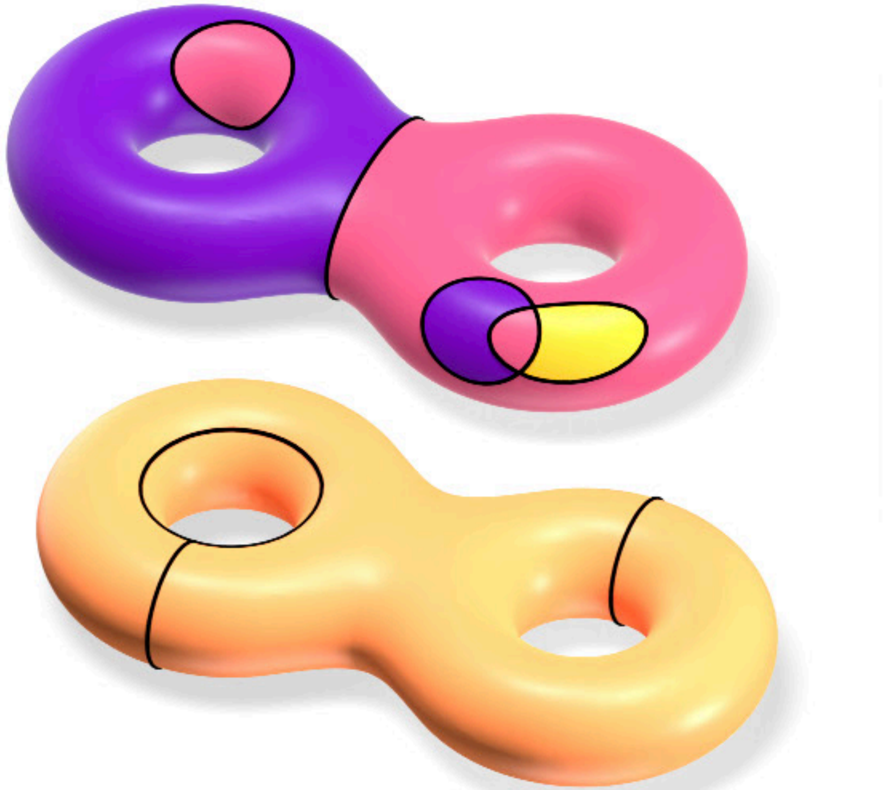
- bounding components that induce valid regions
- nonbounding components.

A closed, completed version of the input curve.



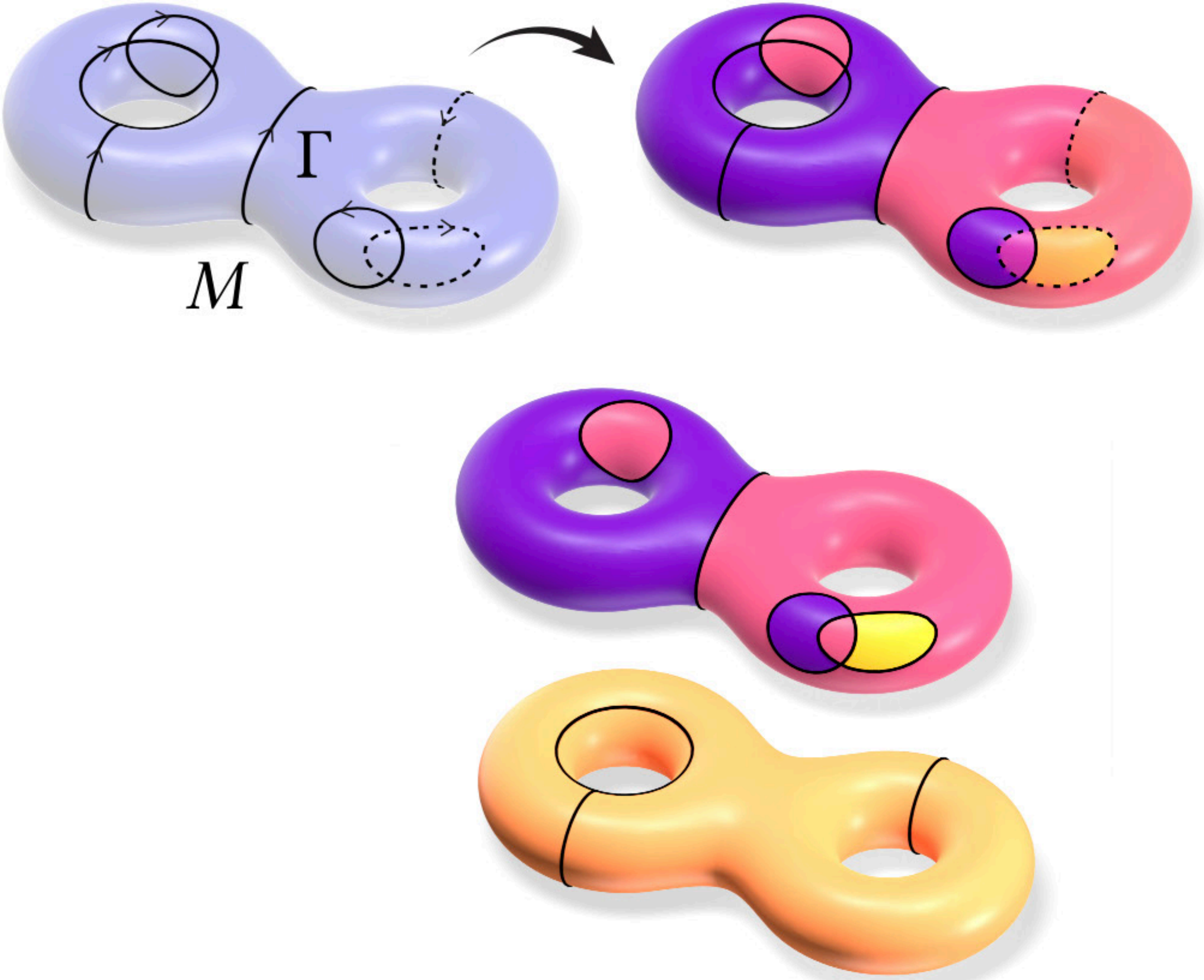
Key insights





Key insights

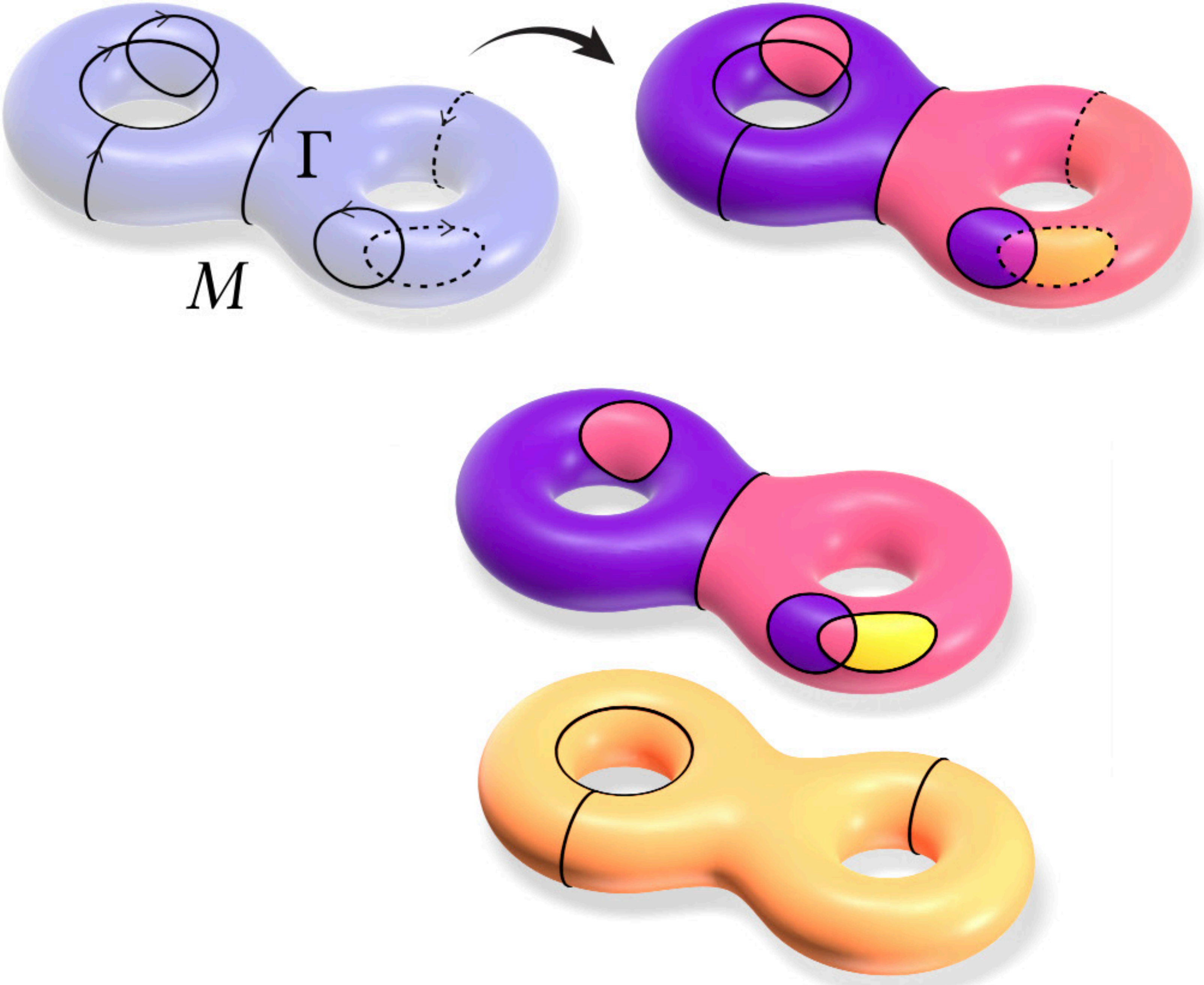
It's difficult to reason about the homology class of *broken* curves.



Key insights

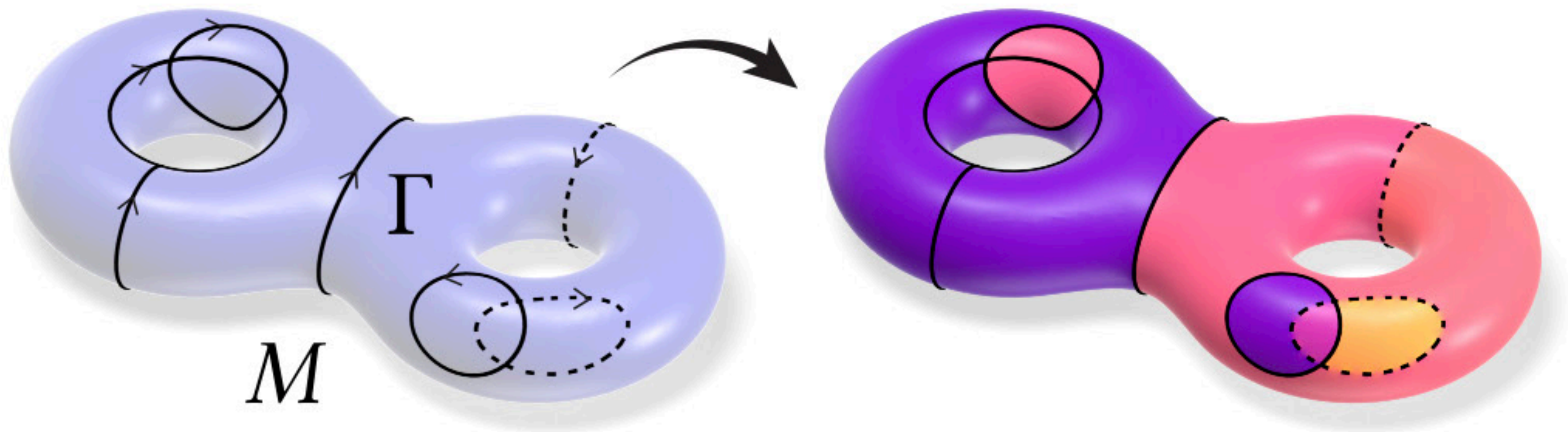
It's difficult to reason about the homology class of *broken* curves.

Instead of processing curves directly, we process functions *dual* to curves using *de Rham cohomology*.



Key insights

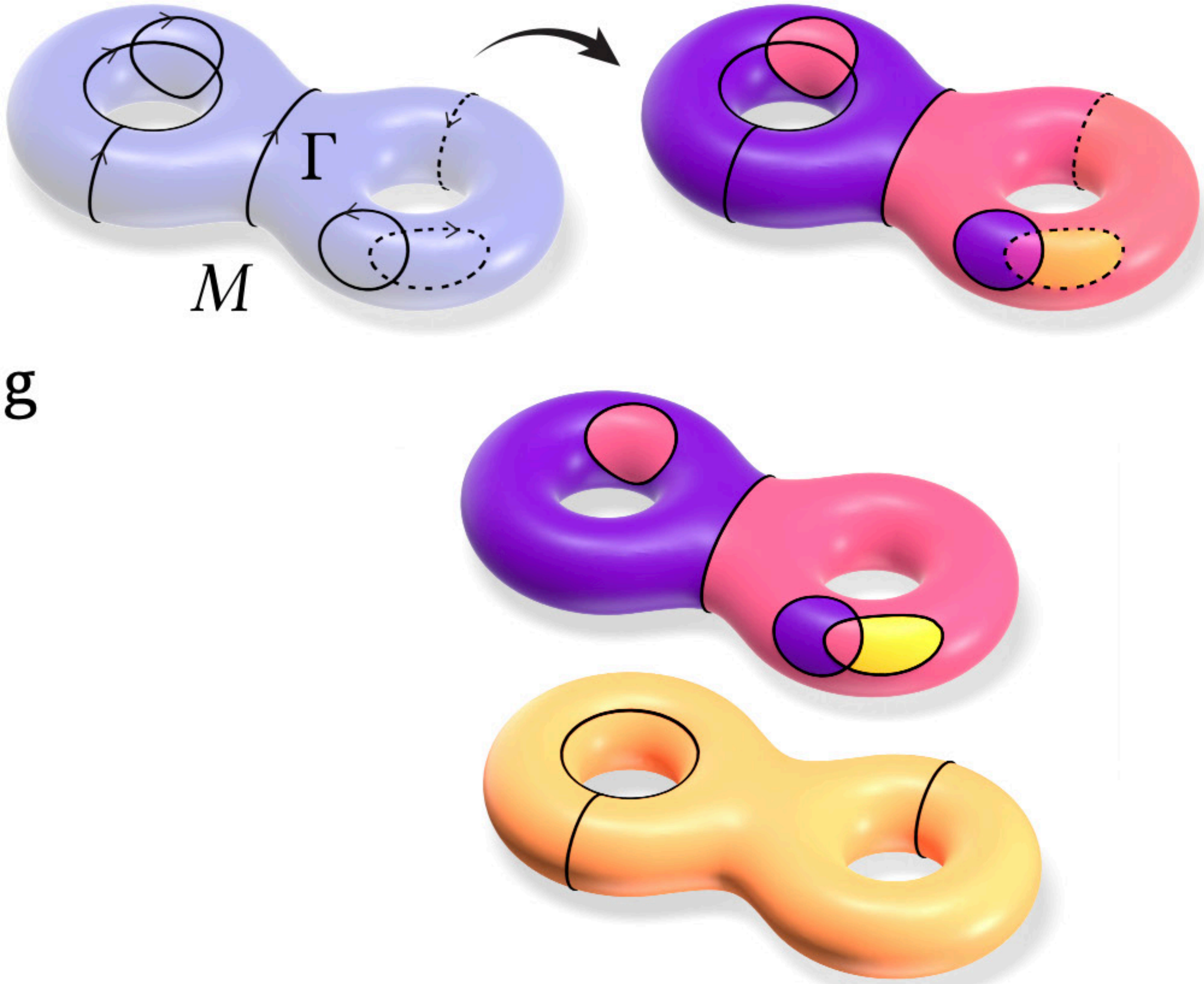
It's difficult to reason about the homology class of *broken* curves.



Instead of processing curves directly, we process functions *dual* to curves using *de Rham cohomology.*

We map from functions back to curves, yielding final output.

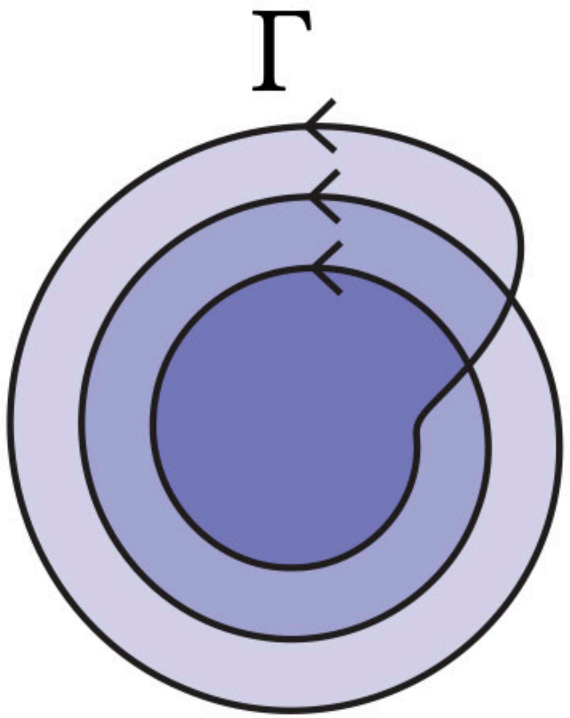
Talk outline



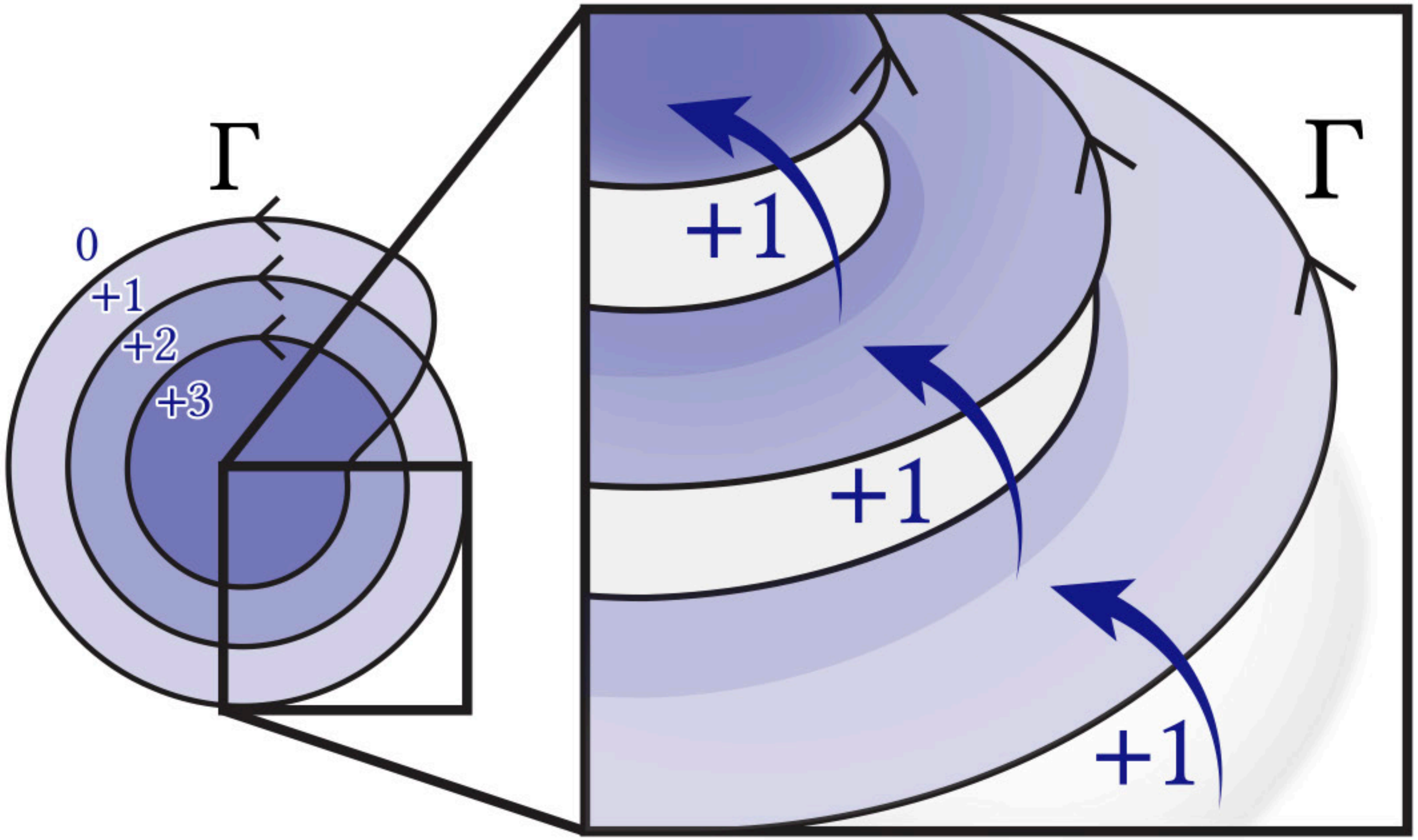
- Algorithm in the smooth setting
- Discretization
- Evaluation & Results

Smooth formulation

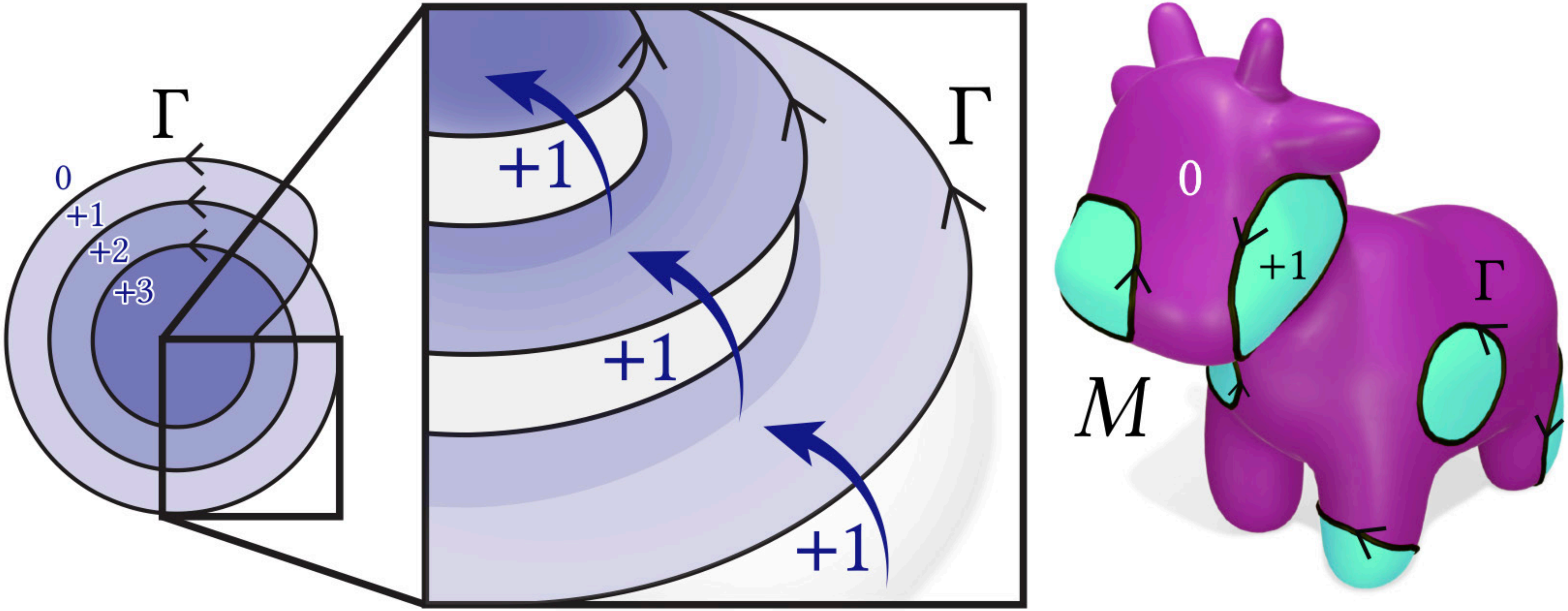
Ordinary winding number: a piecewise constant function



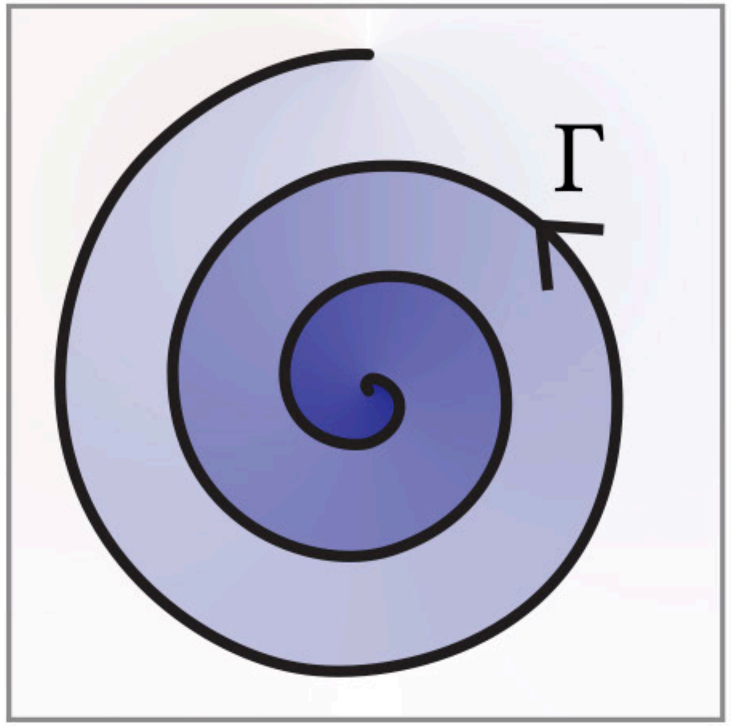
Ordinary winding number: a piecewise constant function



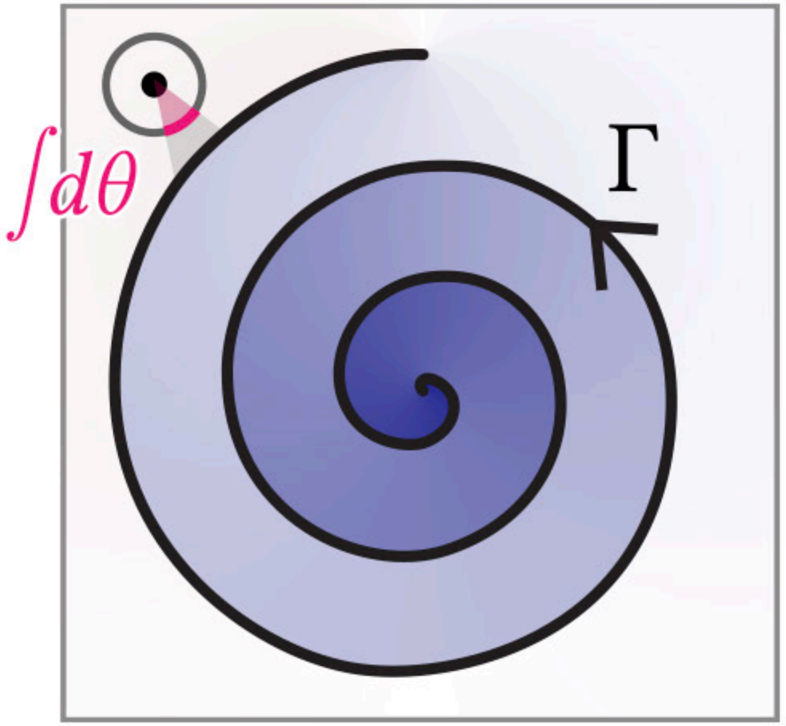
Ordinary winding number: a piecewise constant function



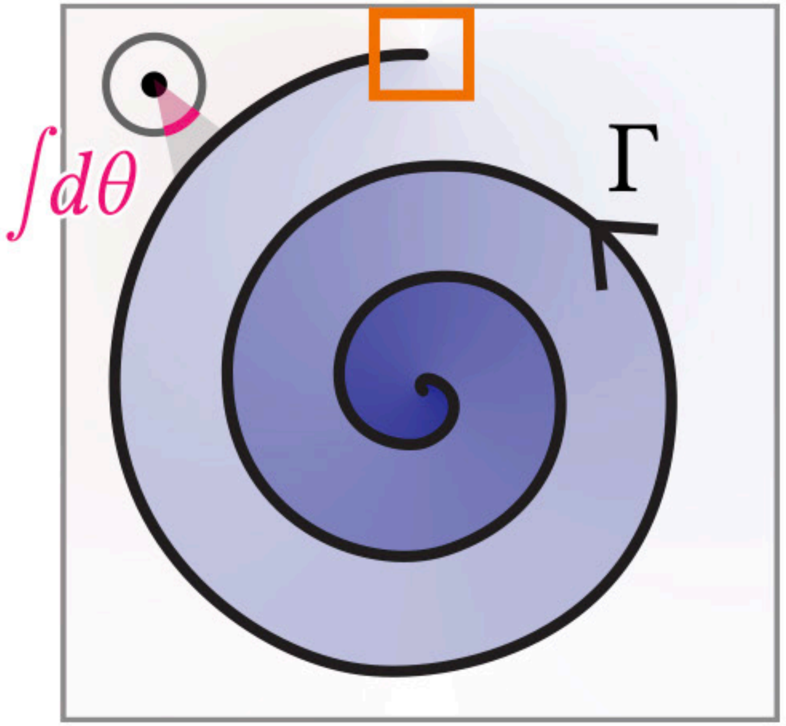
Winding number as a jump harmonic function

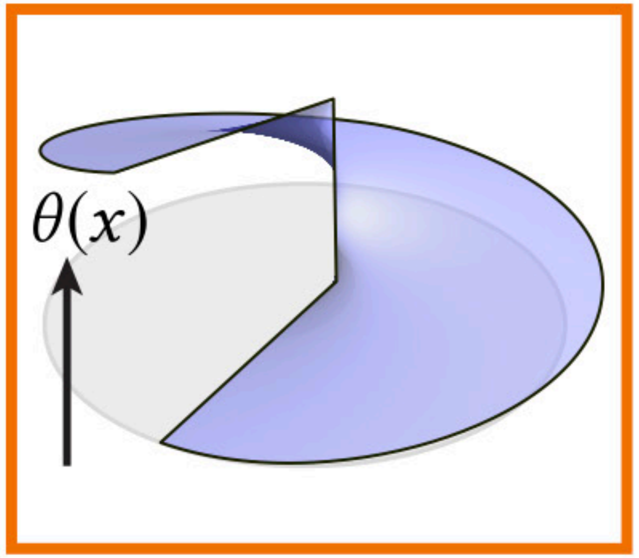


Winding number as a jump harmonic function

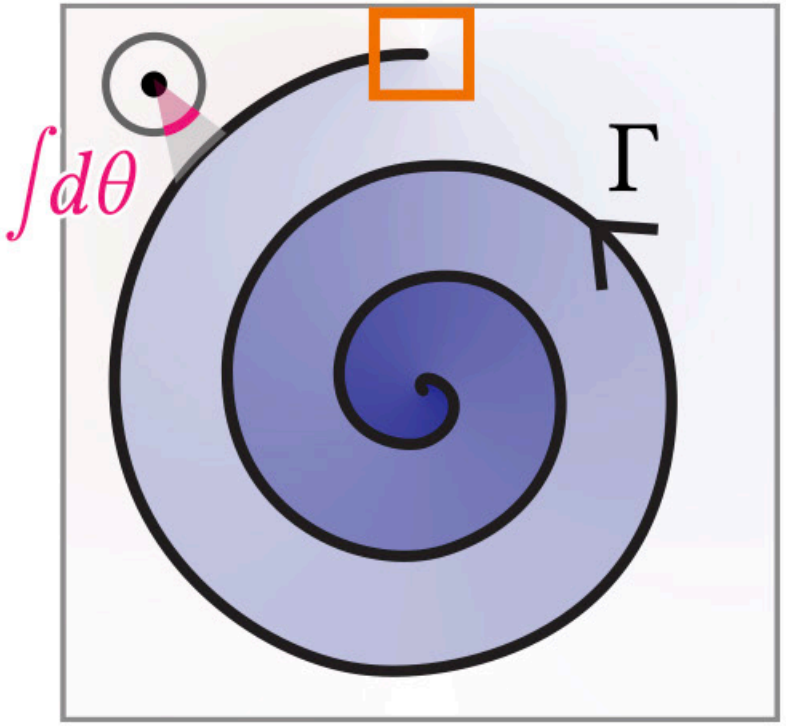


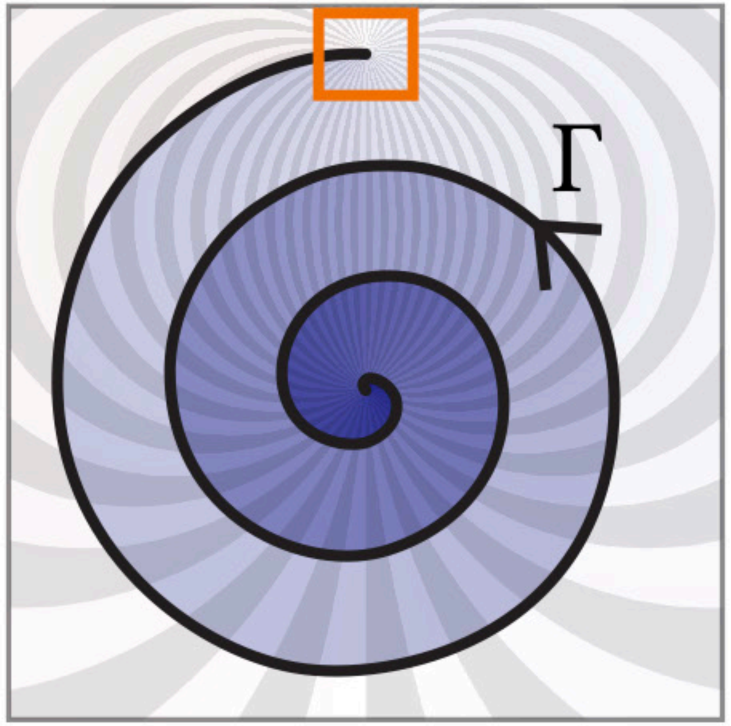
Winding number as a jump harmonic function

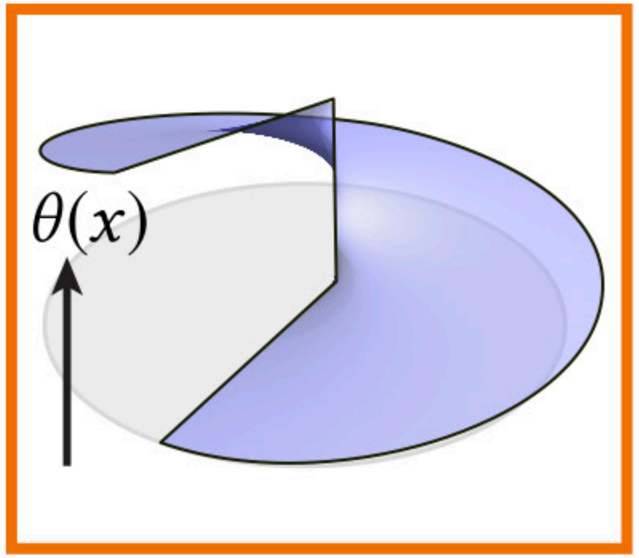


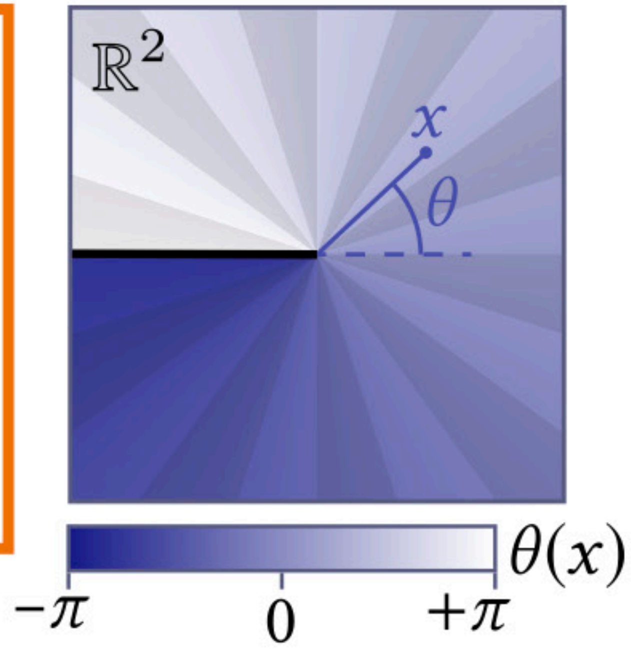


Winding number as an angle-valued function

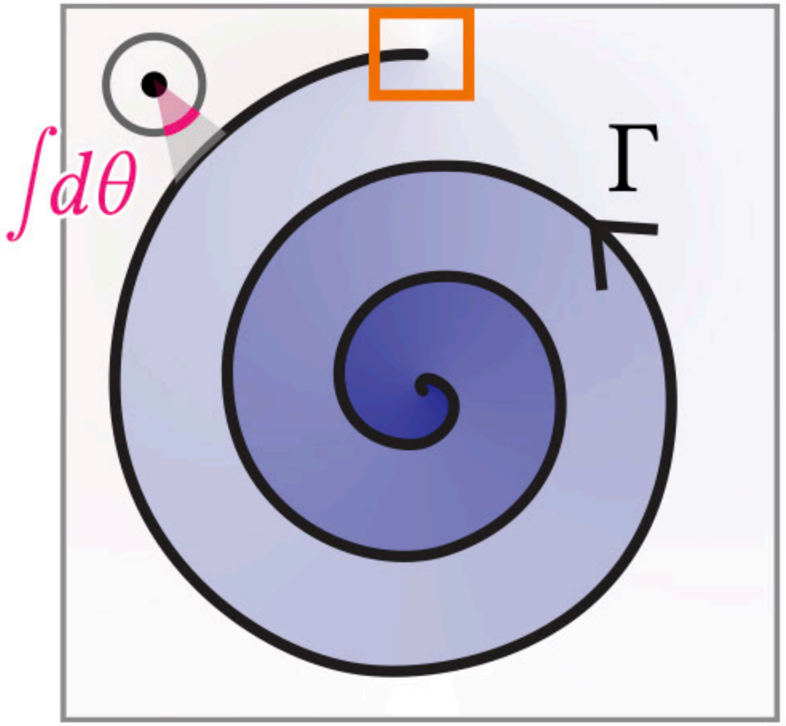


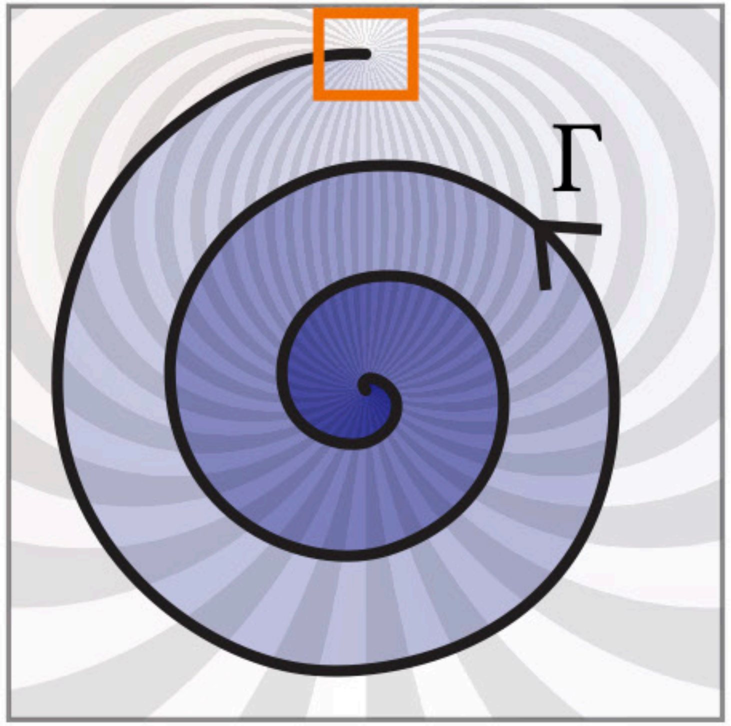


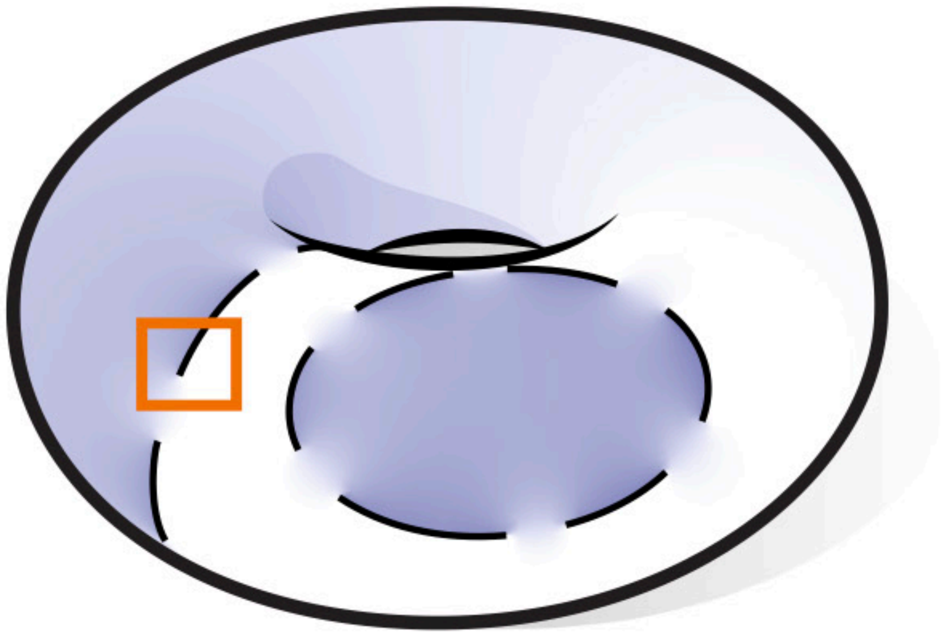


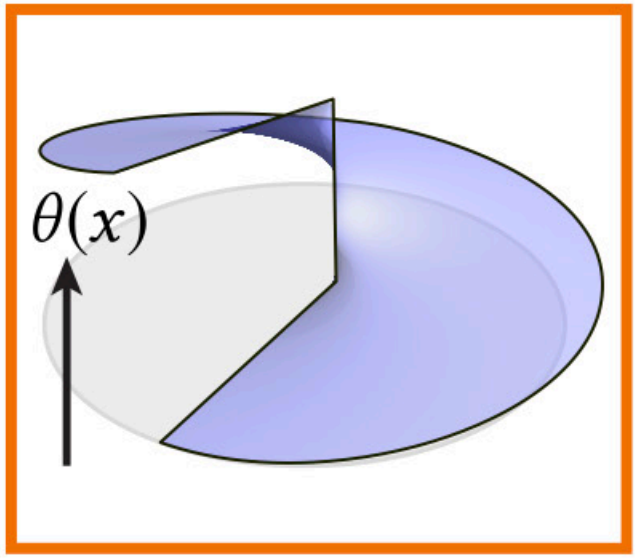


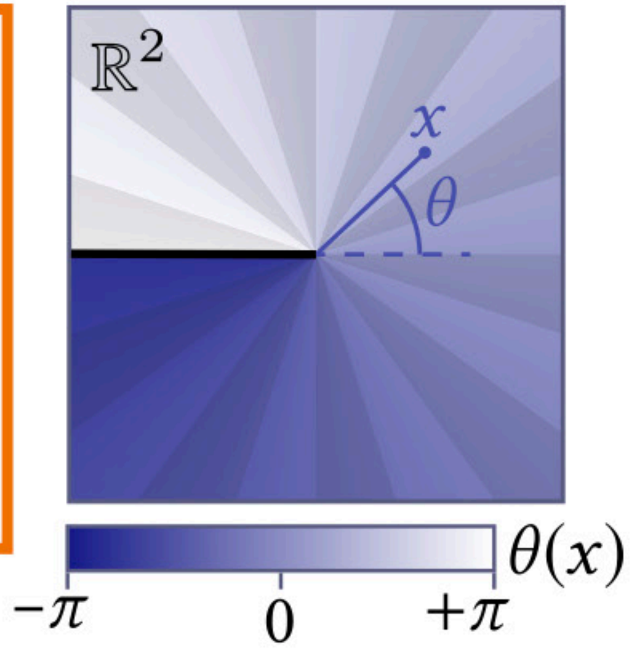
Winding number as an angle-valued function











Winding number as a jump harmonic function

Winding number as a jump harmonic function

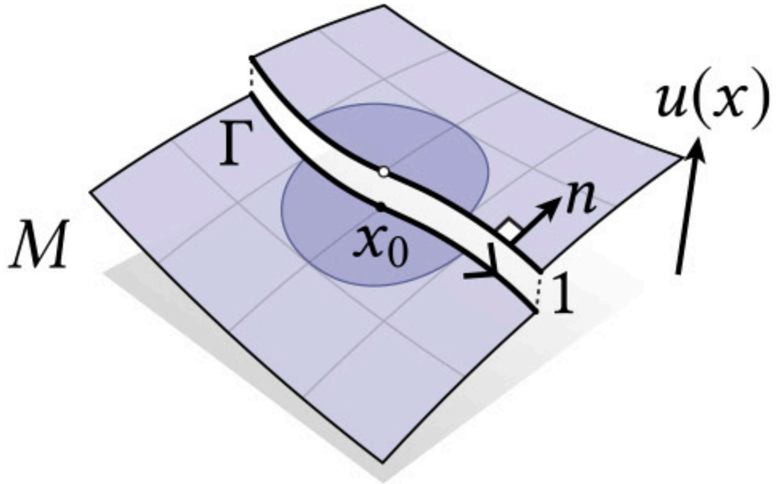
Our starting point is the “Jump Laplace equation”:

$$\Delta u = 0, \qquad \text{on } M \setminus \Gamma,$$

Winding number as a jump harmonic function

Our starting point is the "Jump Laplace equation":

$$\begin{aligned} \Delta u &= 0, && \text{on } M \setminus \Gamma, \\ u^+ - u^- &= 1,^* && \text{on } \Gamma, \end{aligned}$$

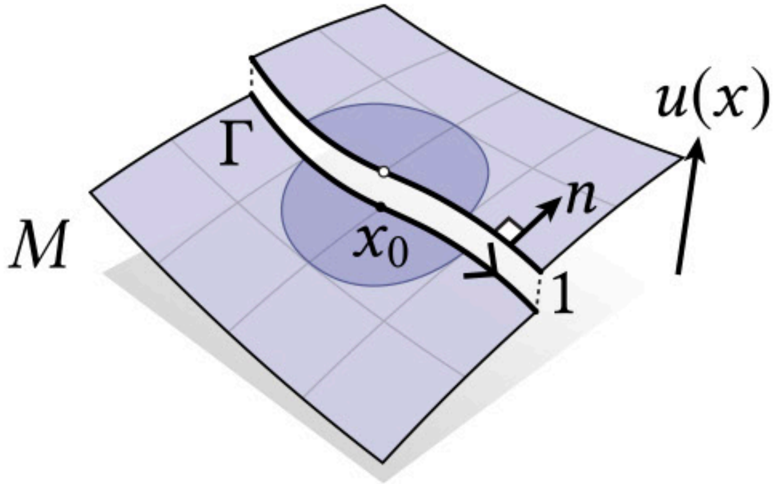


*In general, $u^+(x) - u^-(x)$ = number of times Γ covers x.

Winding number as a jump harmonic function

Our starting point is the "Jump Laplace equation":

$$\begin{aligned} \Delta u &= 0, && \text{on } M \setminus \Gamma, \\ u^+ - u^- &= 1^*, && \text{on } \Gamma, \\ \partial u^+/\partial n &= \partial u^-/\partial n, && \text{on } \Gamma. \end{aligned}$$

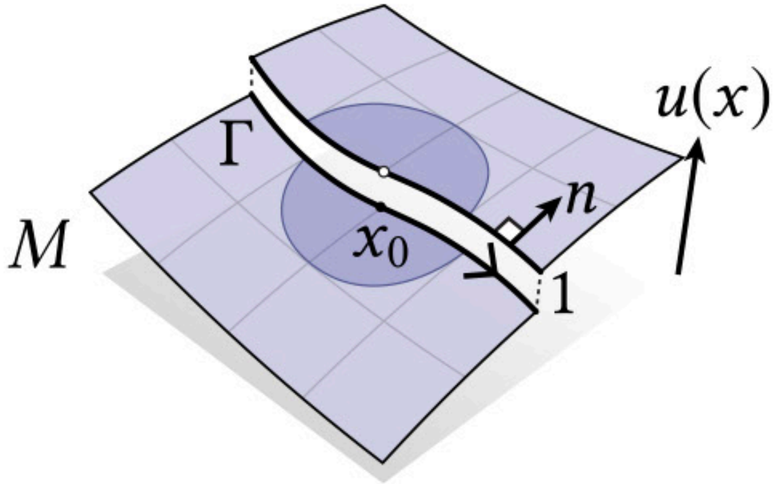


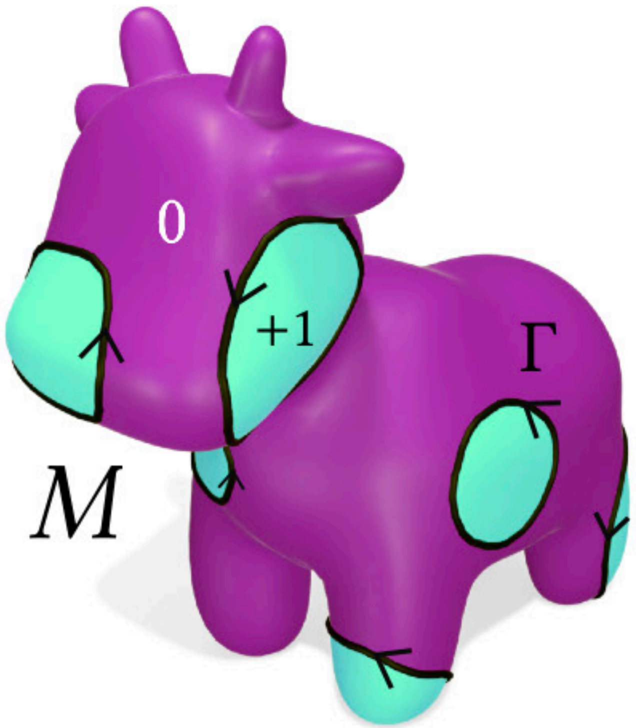
*In general, $u^+(x) - u^-(x)$ = number of times Γ covers x.

Winding number as a jump harmonic function

Our starting point is the "Jump Laplace equation":

$$
\begin{aligned}
\Delta u &= 0, && \text{on } M \setminus \Gamma, \\
u^+ - u^- &= 1^*, && \text{on } \Gamma, \\
\partial u^+ / \partial n &= \partial u^- / \partial n, && \text{on } \Gamma.
\end{aligned}
$$



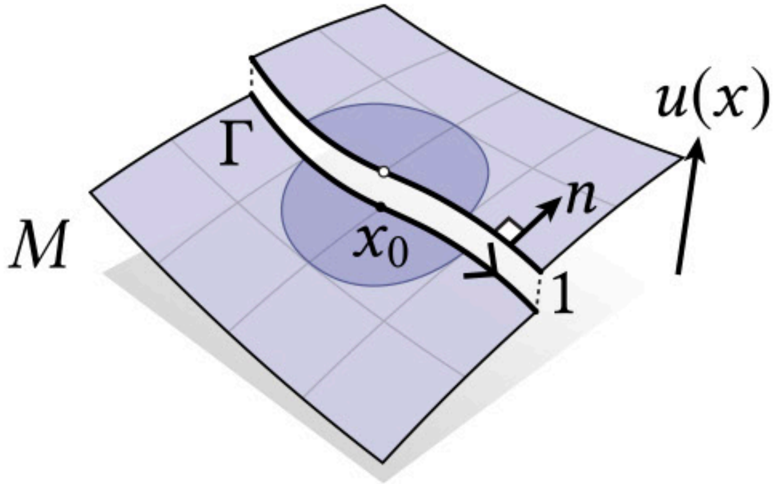


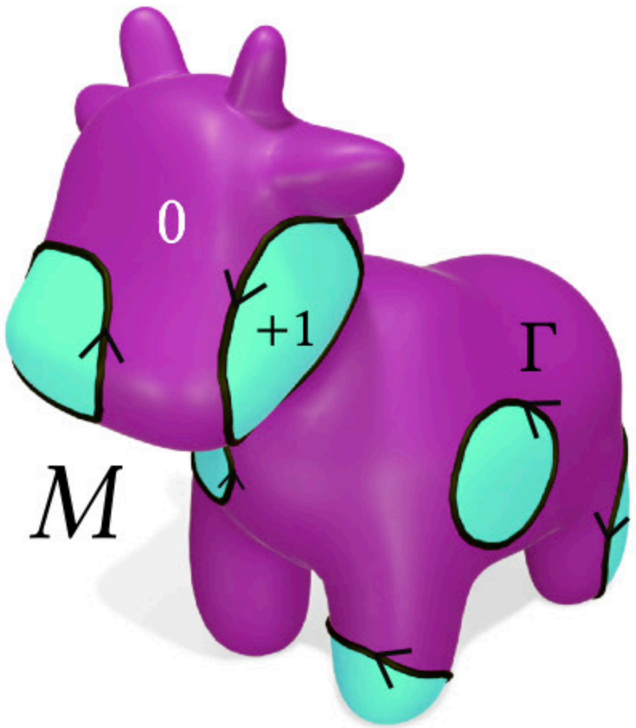
*In general, $u^+(x) - u^-(x) =$ number of times Γ covers x.

Winding number as a jump harmonic function

Our starting point is the "Jump Laplace equation":

$$\begin{aligned} \Delta u &= 0, && \text{on } M \setminus \Gamma, \\ u^+ - u^- &= 1,^* && \text{on } \Gamma, \\ \partial u^+/\partial n &= \partial u^-/\partial n, && \text{on } \Gamma. \end{aligned}$$



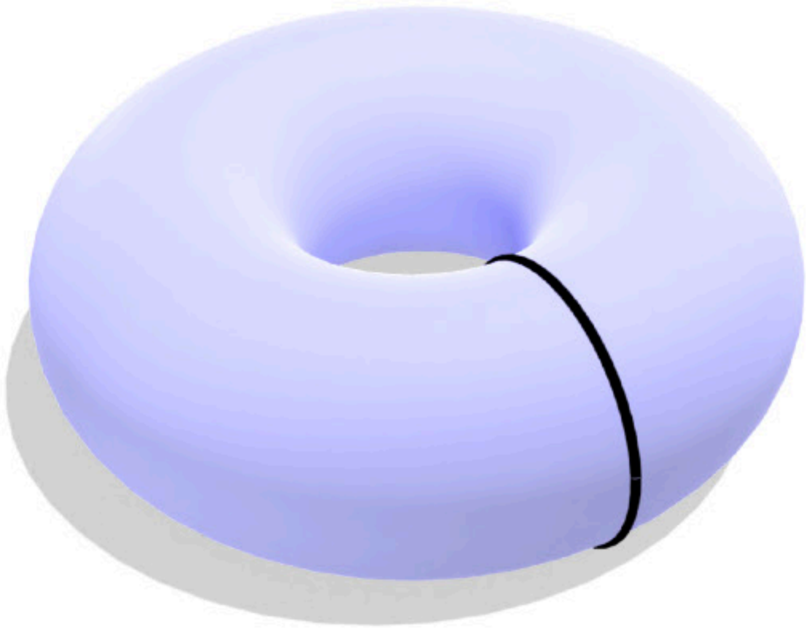


If M is simply-connected, and all curves are closed, then we're done!

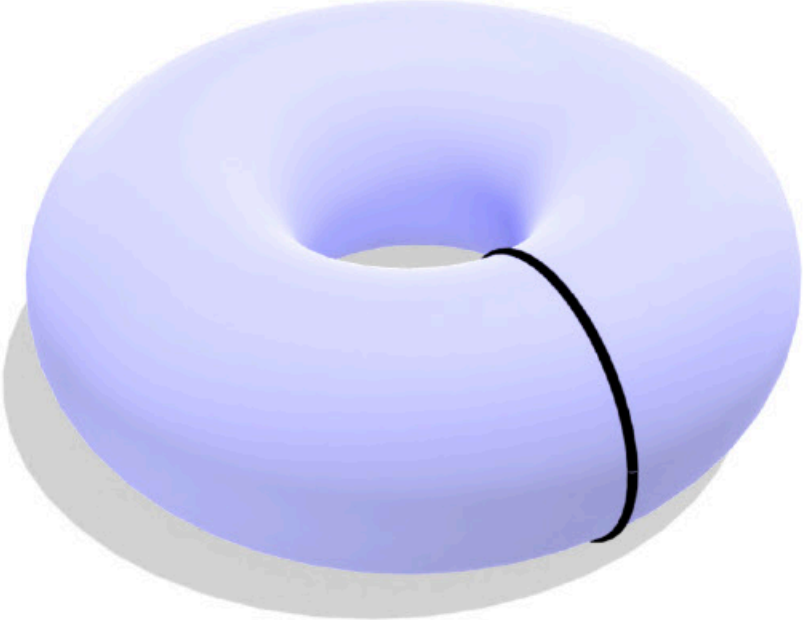
*In general, $u^+(x) - u^-(x)$ = number of times Γ covers x.

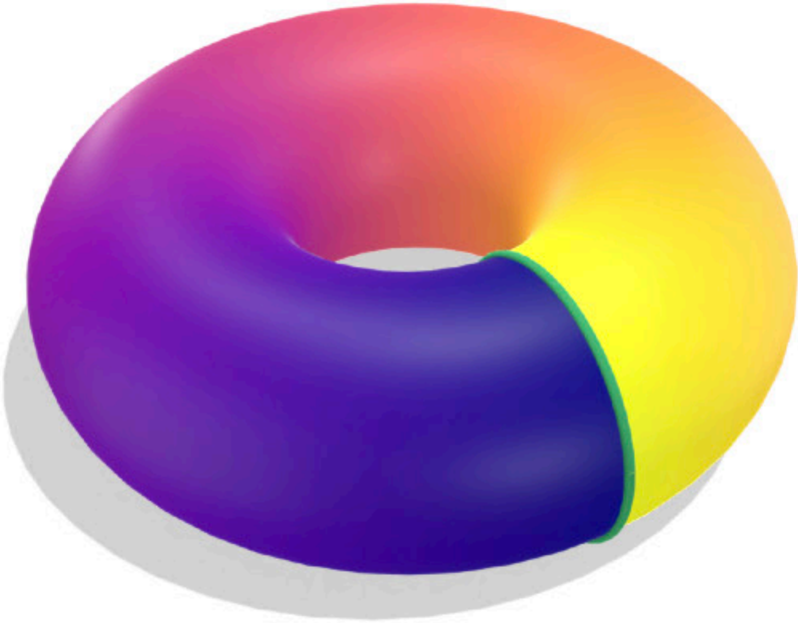
Nonbounding loops complicate region labeling

input

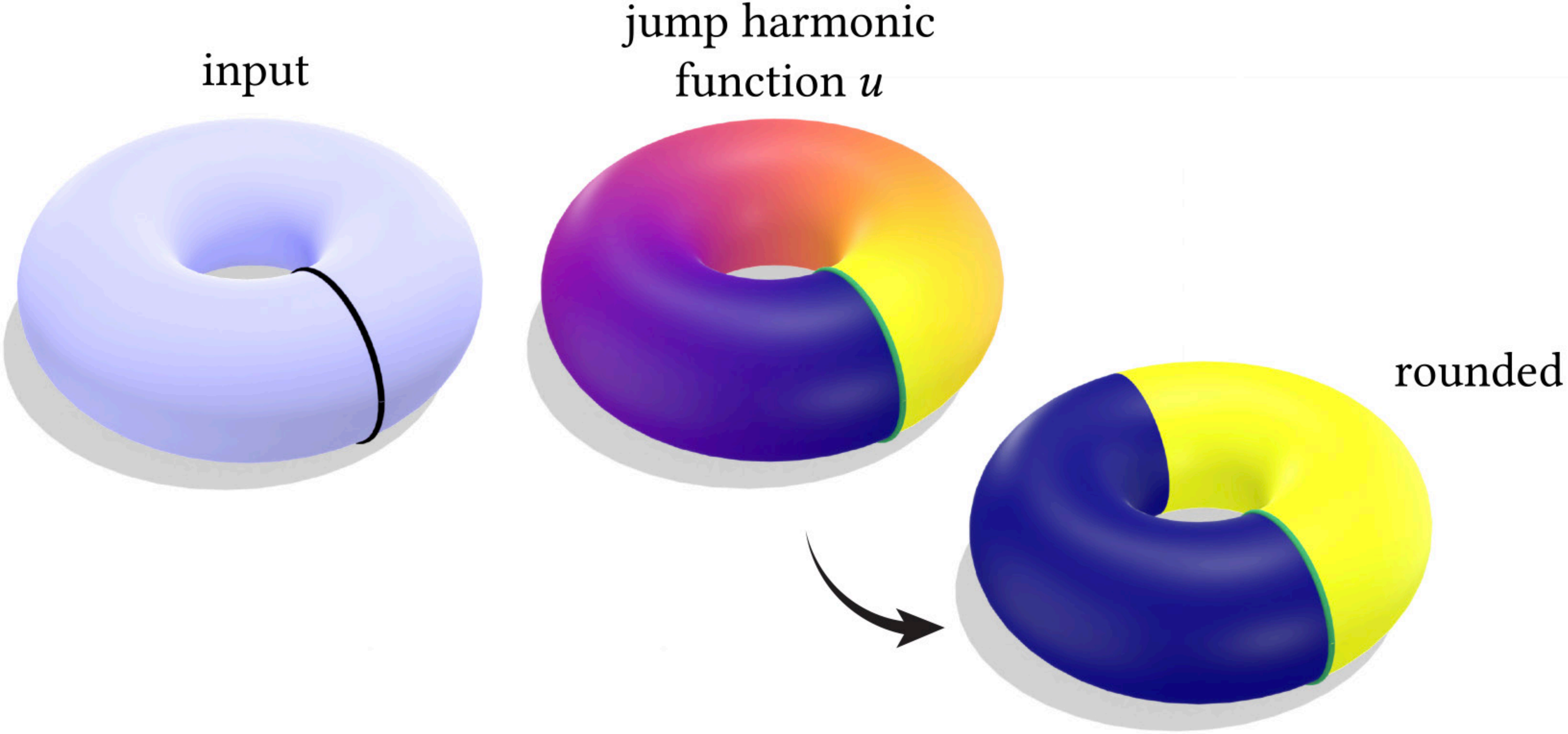


Nonbounding loops complicate region labeling

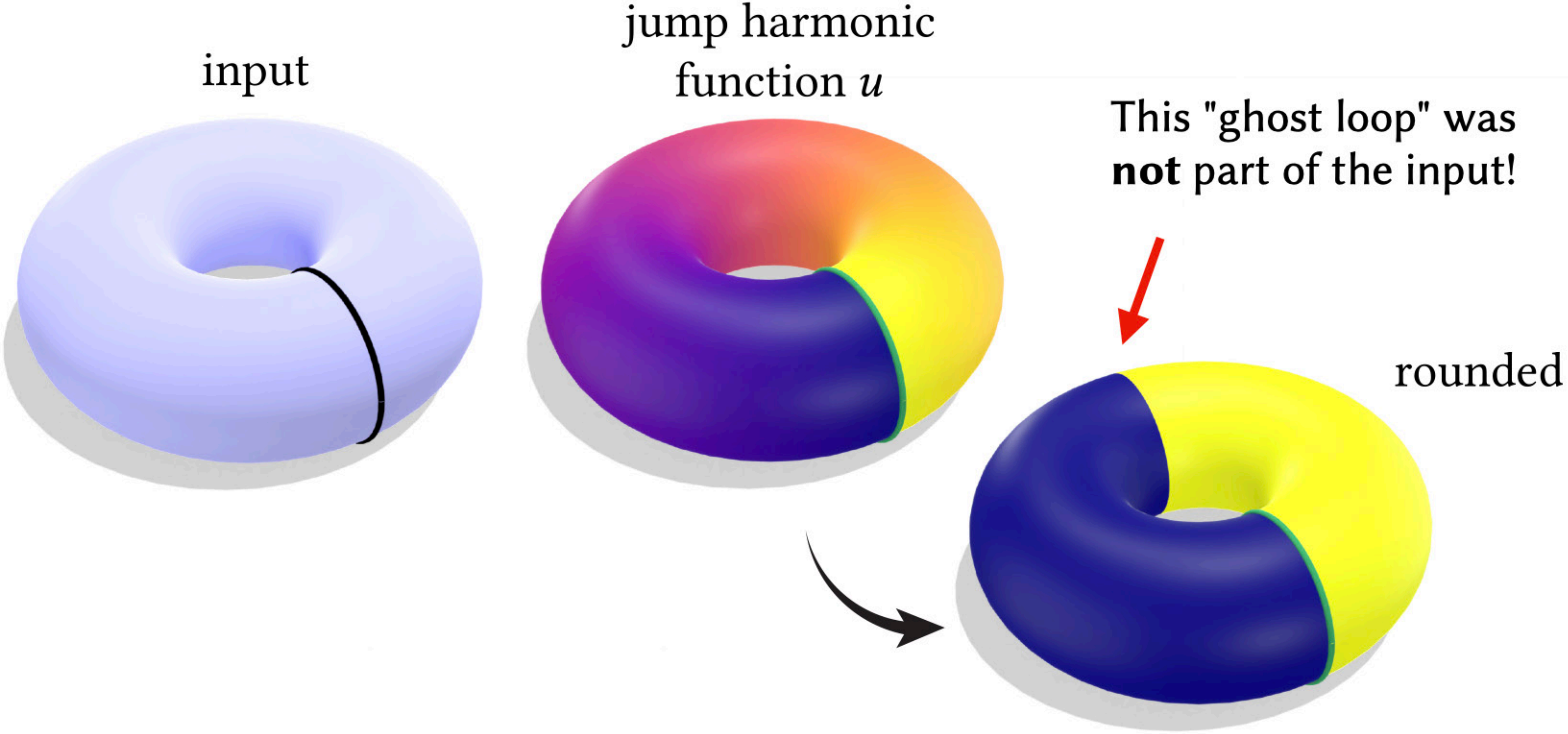




Nonbounding loops complicate region labeling



Nonbounding loops complicate region labeling



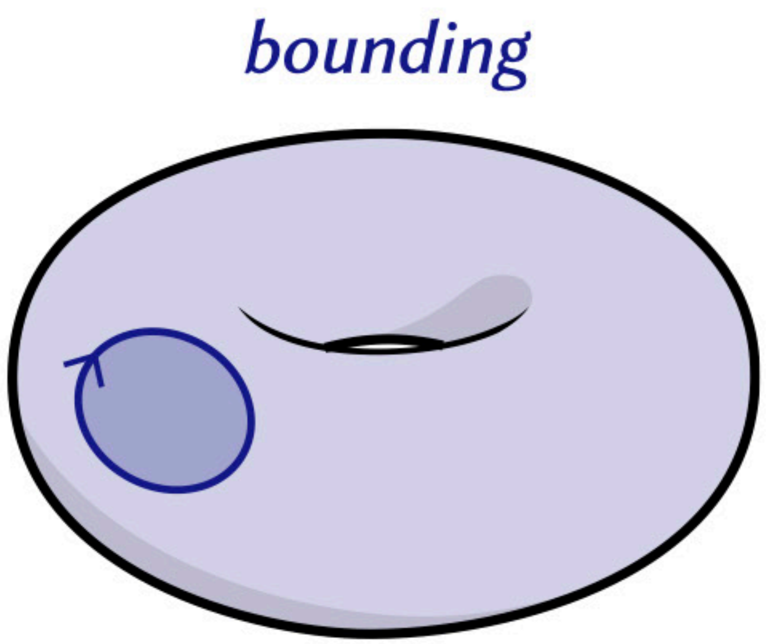
What are nonbounding loops?

What are nonbounding loops?

The *first homology group* $H_1(M) = \ker(\partial_1) \setminus \operatorname{im}(\partial_2)$ tells us about (closed) curves that are not boundaries of regions.

What are nonbounding loops?

The *first homology group* $H_1(M) = \ker(\partial_1) \setminus \mathrm{im}(\partial_2)$ tells us about (closed) curves that are not boundaries of regions.



For clarity,

bounding := *nullhomologous*

What are nonbounding loops?

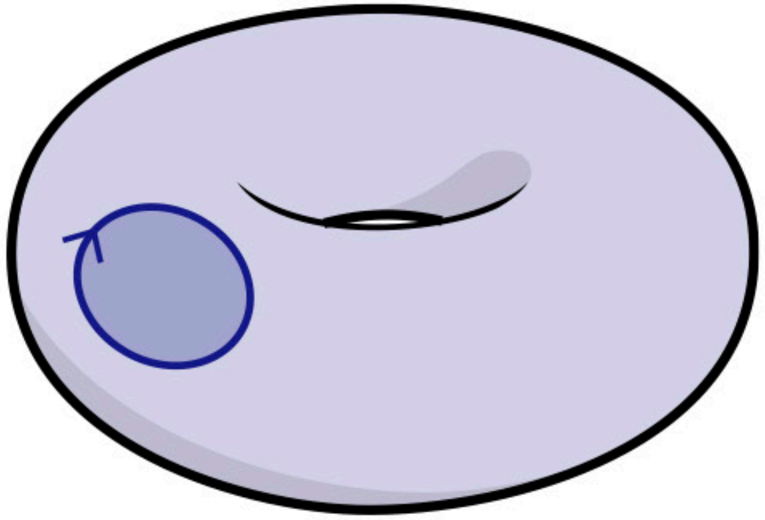
The *first homology group* $H_1(M) = \ker(\partial_1) \setminus \operatorname{im}(\partial_2)$ tells us about (closed) curves that are not boundaries of regions.

For clarity,

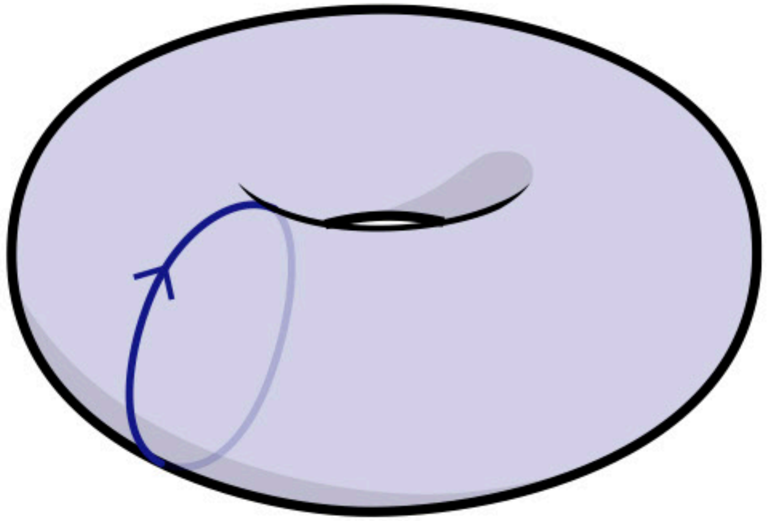
bounding := *nullhomologous*

nonbounding := *non-nullhomologous*

bounding



nonbounding



How does homology apply to broken curves?

How does homology apply to broken curves?

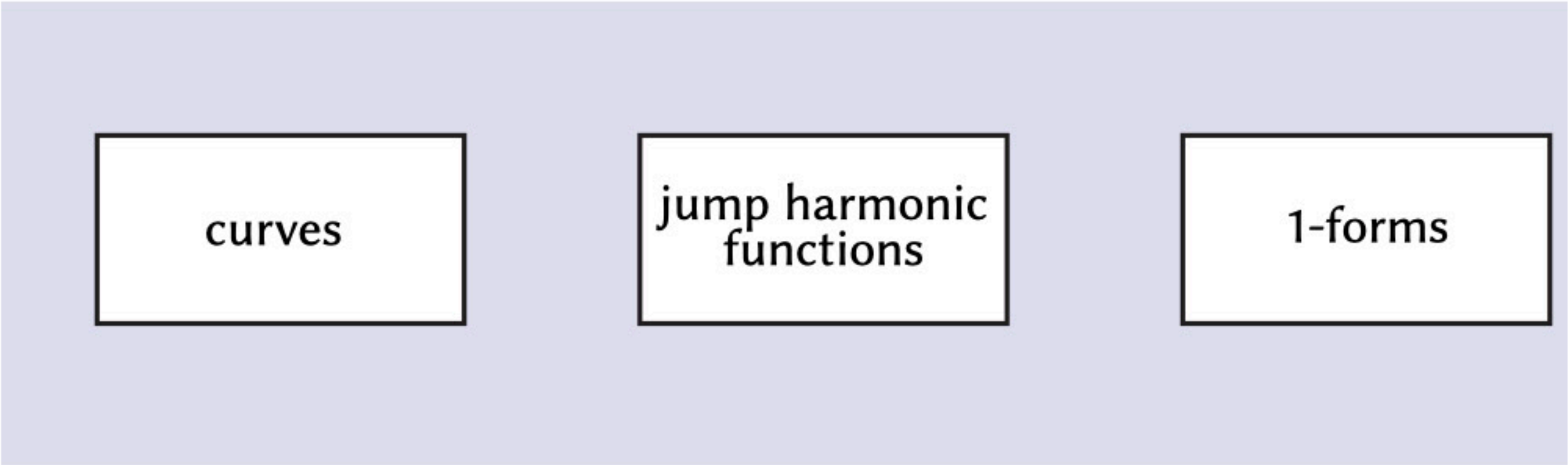
It’s difficult to reason about curves directly.

How does homology apply to broken curves?

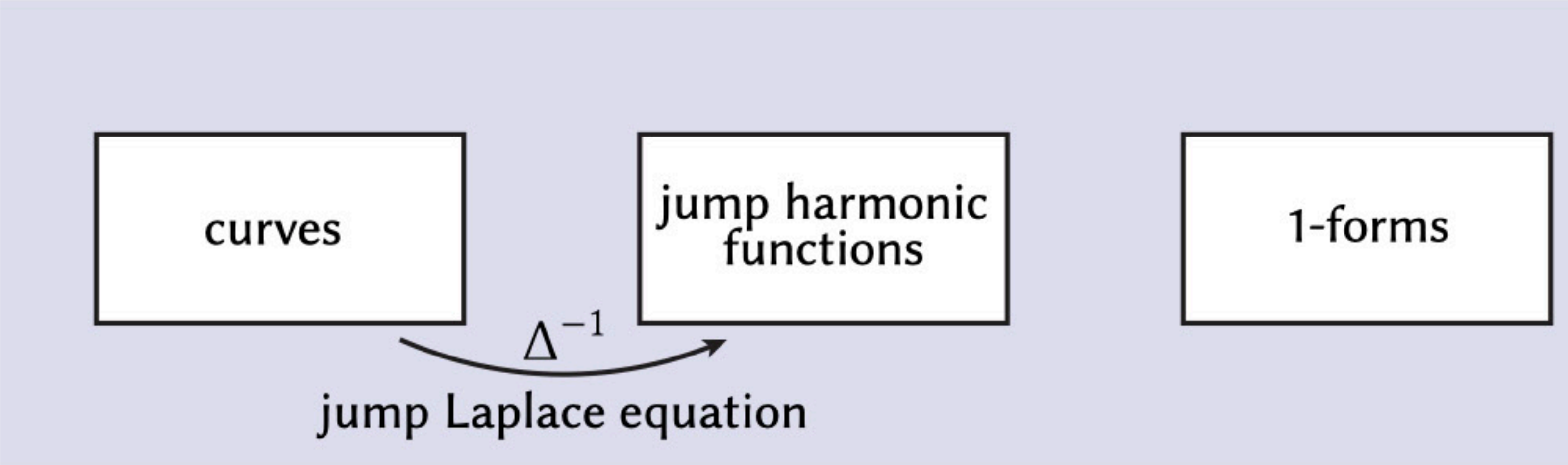
It's difficult to reason about curves directly.

Instead of processing curves directly, we process functions *dual* to curves using *de Rham cohomology.*

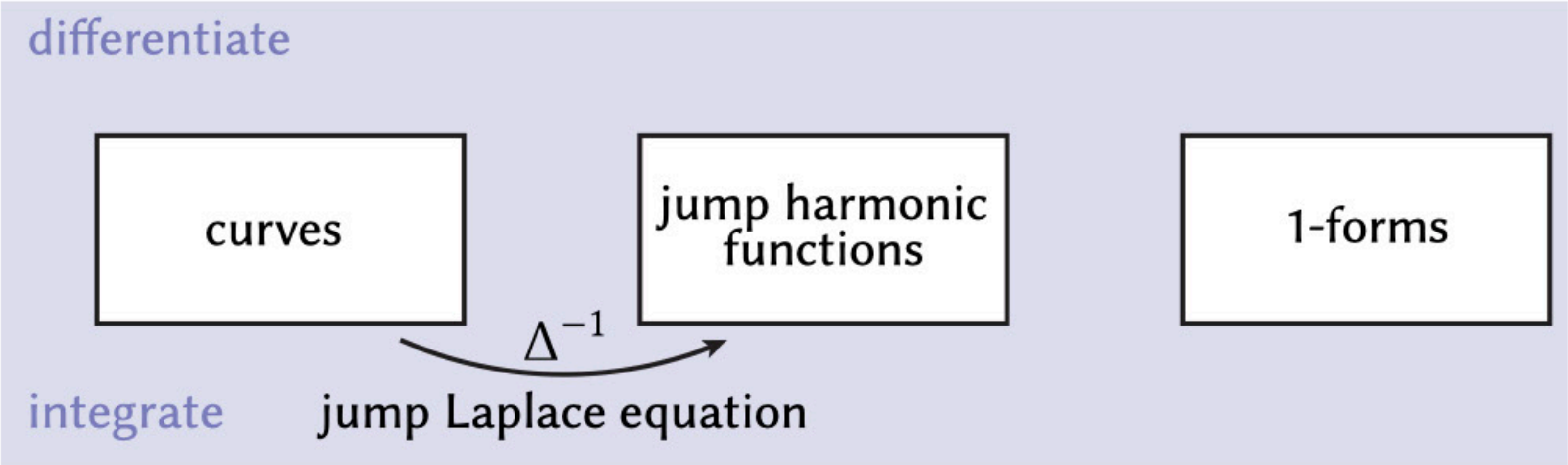
A preview of the journey ahead



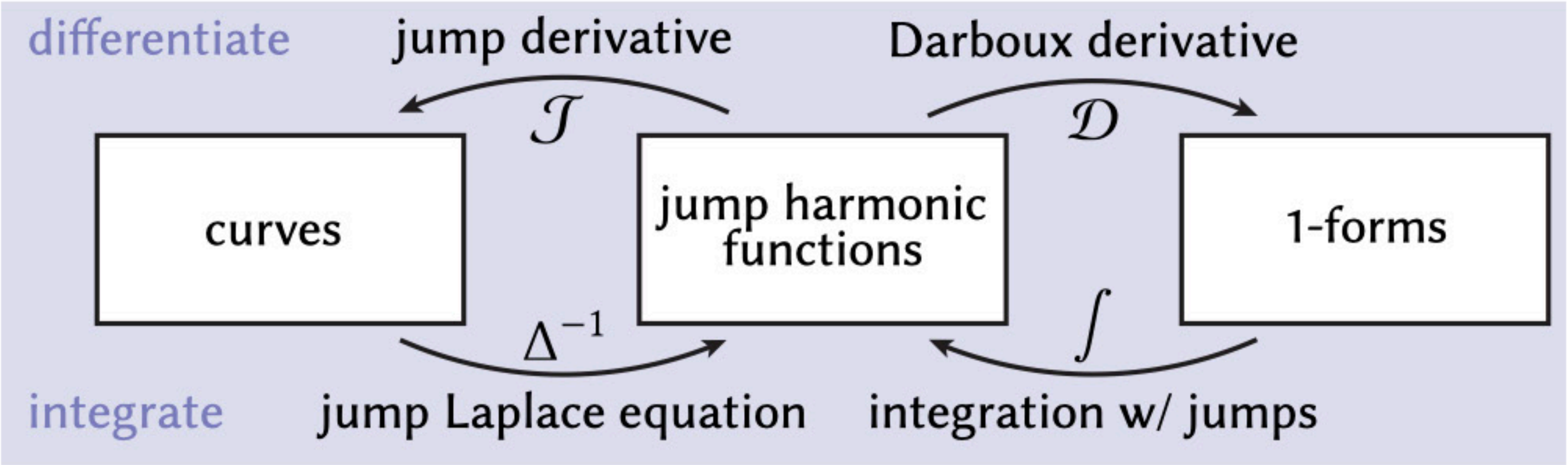
A preview of the journey ahead

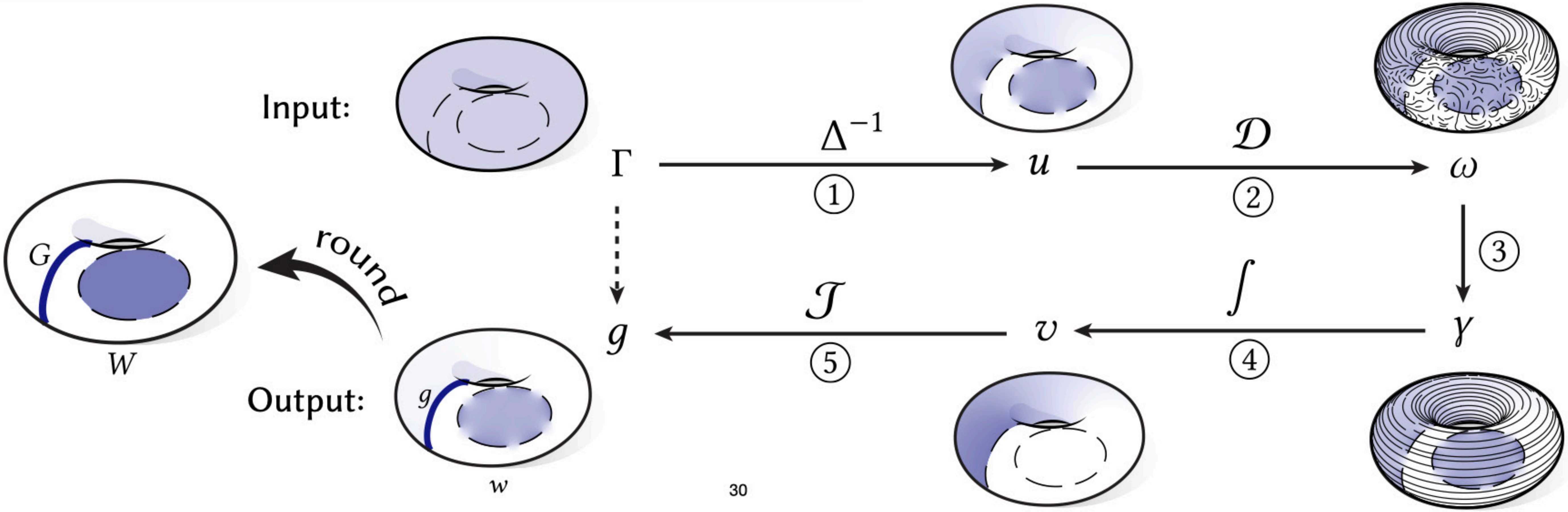


A preview of the journey ahead

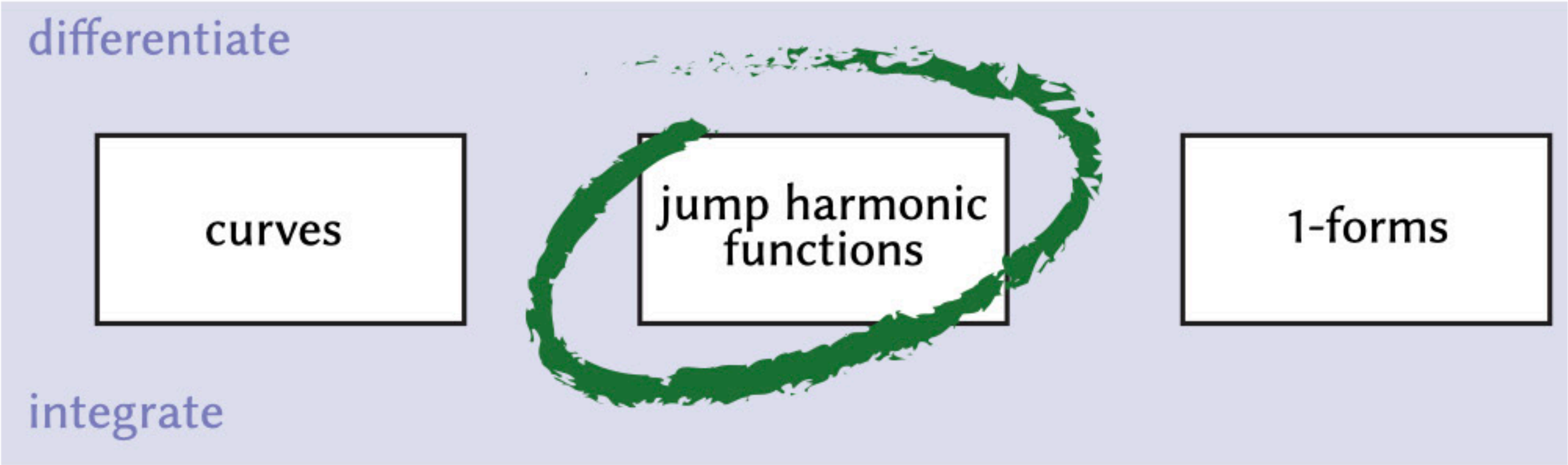


A preview of the journey ahead





Jump harmonic functions

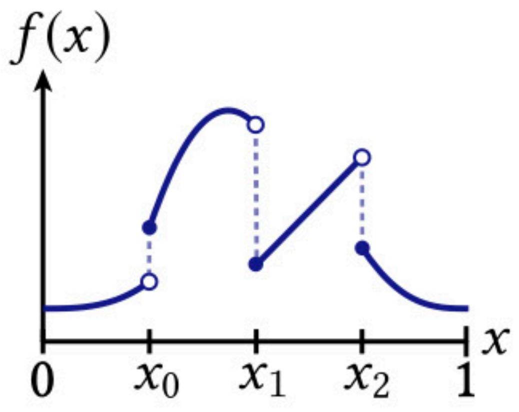


First let's talk about differentiating & integrating jump harmonic functions.

Derivatives of discontinuous functions

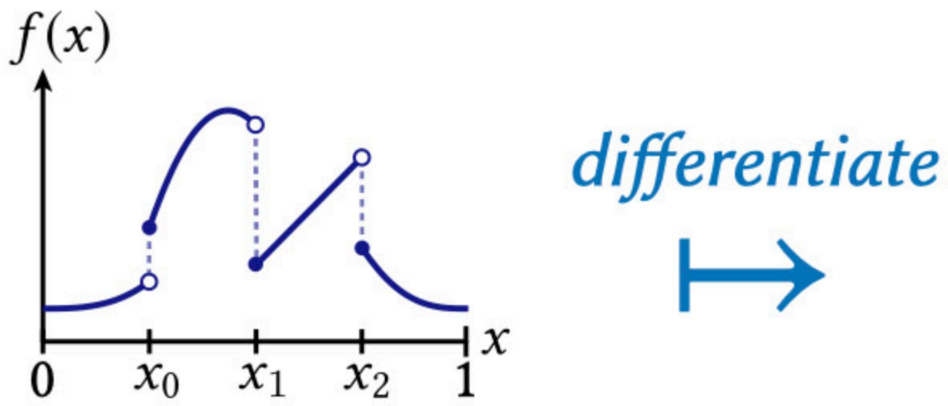
Derivatives of discontinuous functions

Consider a periodic 1D function $f(x)$ on [0,1]:



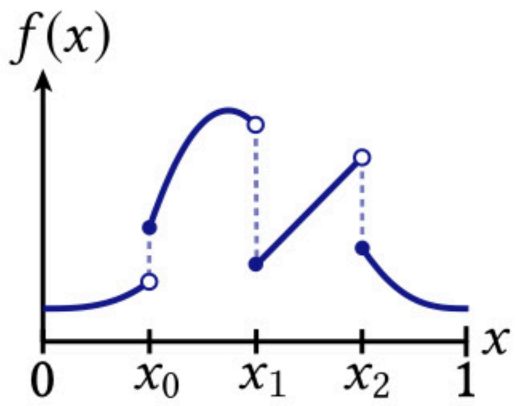
Derivatives of discontinuous functions

Consider a periodic 1D function $f(x)$ on [0,1]:



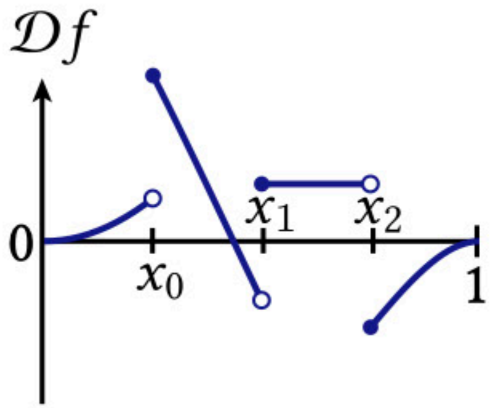
Derivatives of discontinuous functions

Consider a periodic 1D function $f(x)$ on [0,1]:



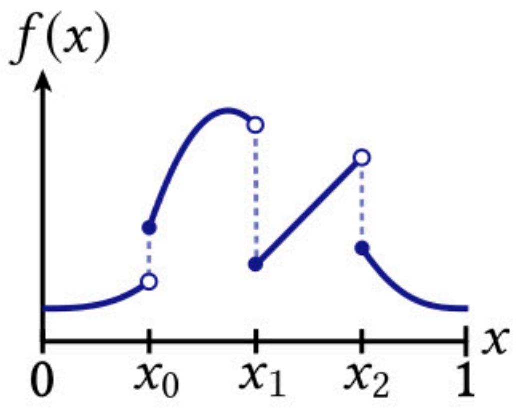
differentiate $\mapsto$

continuous part



Derivatives of discontinuous functions

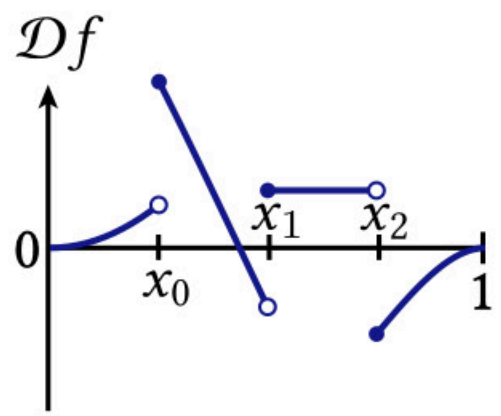
Consider a periodic 1D function $f(x)$ on [0,1]:



differentiate

$\mapsto$

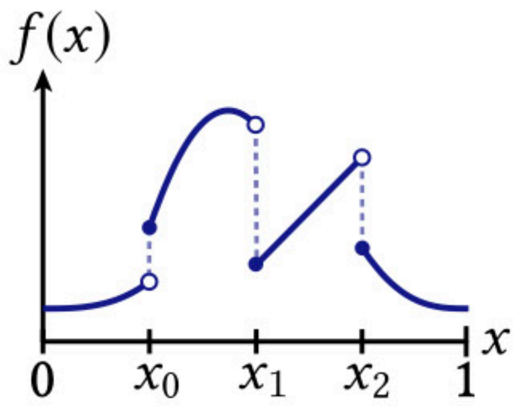
continuous part



$$\omega := \mathcal{D}f$$

Derivatives of discontinuous functions

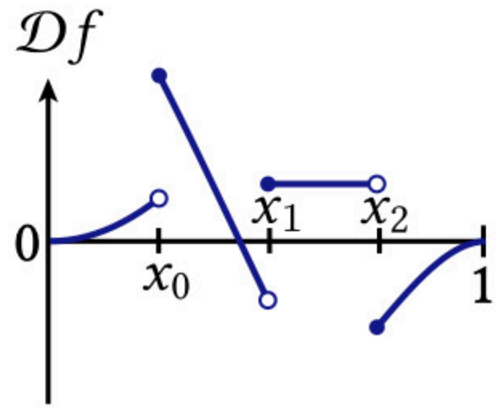
Consider a periodic 1D function $f(x)$ on [0,1]:



differentiate

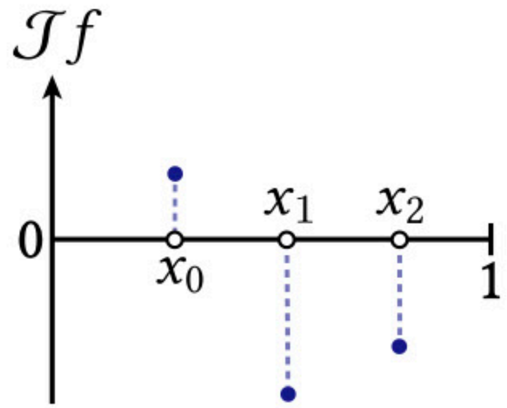
$\mapsto$

continuous part



+

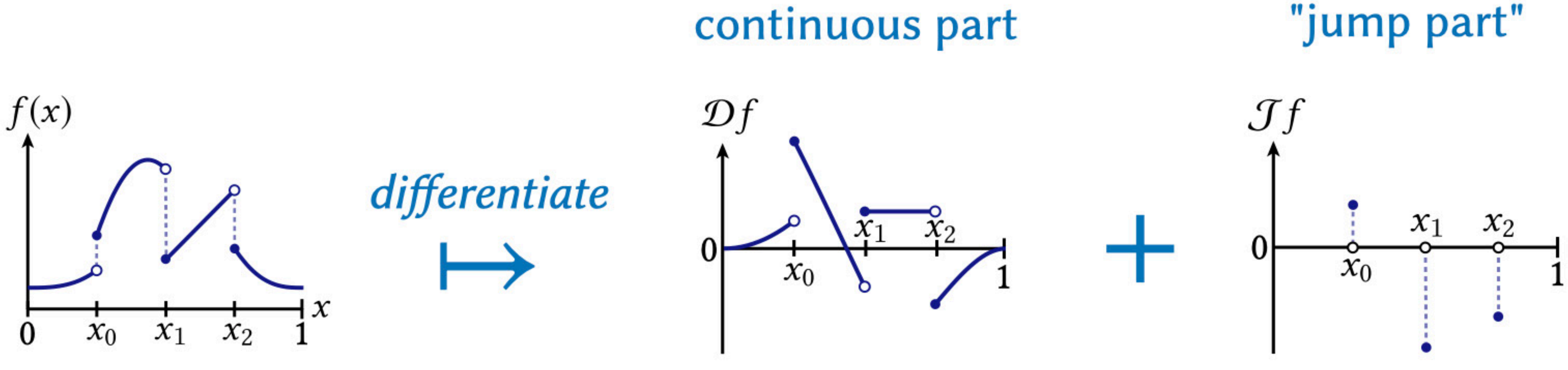
"jump part"



$$\omega := \mathcal{D}f$$

Derivatives of discontinuous functions

Consider a periodic 1D function $f(x)$ on [0,1]:



$\omega := \mathcal{D}f$

$\mathcal{J}f = \Sigma_i \Lambda_i \delta_{x_i}$

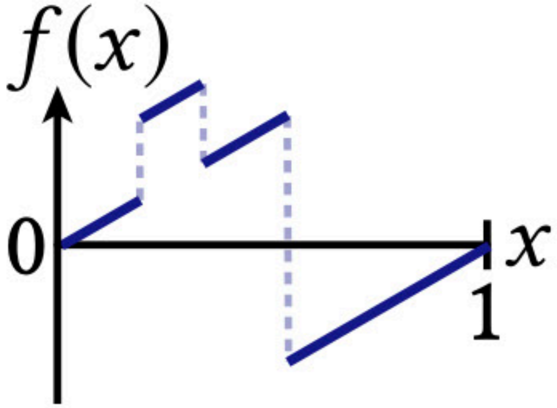
Derivatives of jump harmonic functions

For a **jump harmonic** function f, the *Darboux derivative* $\omega := \mathcal{D}f$ "forgets" jumps:

Derivatives of jump harmonic functions

For a **jump harmonic** function *f*, the *Darboux derivative* $\omega := \mathcal{D}f$ "forgets" jumps:

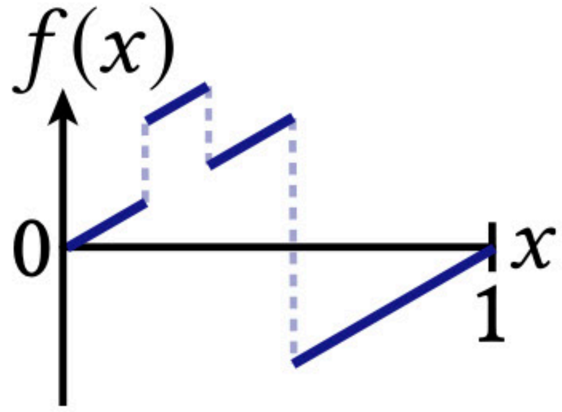
jump harmonic function



Derivatives of jump harmonic functions

For a **jump harmonic** function f, the *Darboux derivative* $\omega := \mathcal{D}f$ "forgets" jumps:

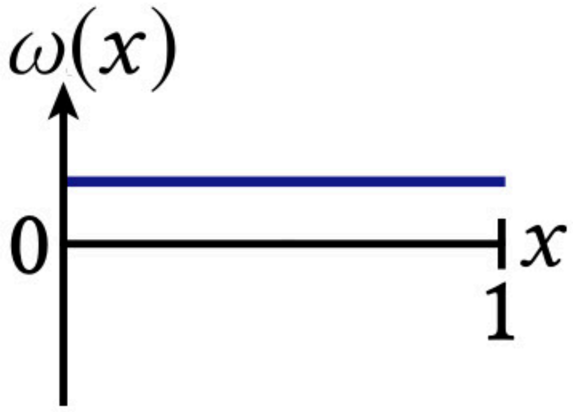
jump harmonic function



differentiate

$\mapsto$

Darboux derivative

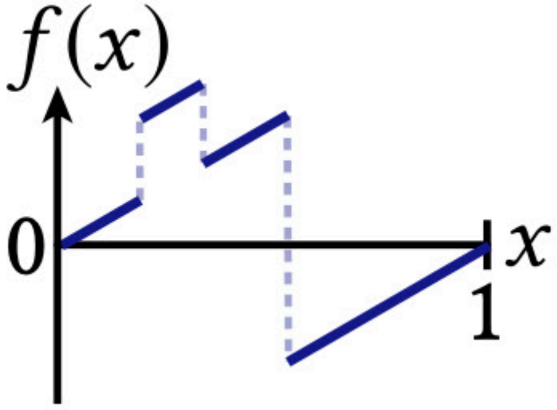


$\omega := \mathcal{D}f$

Derivatives of jump harmonic functions

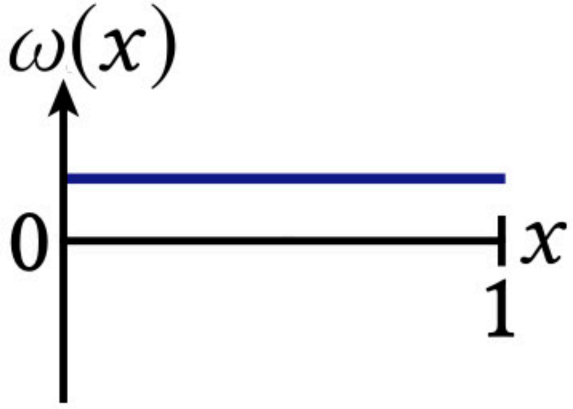
For a **jump harmonic** function *f*, the *Darboux derivative* $\omega := \mathcal{D}f$ "forgets" jumps:

jump harmonic function



differentiate $\mapsto$

Darboux derivative



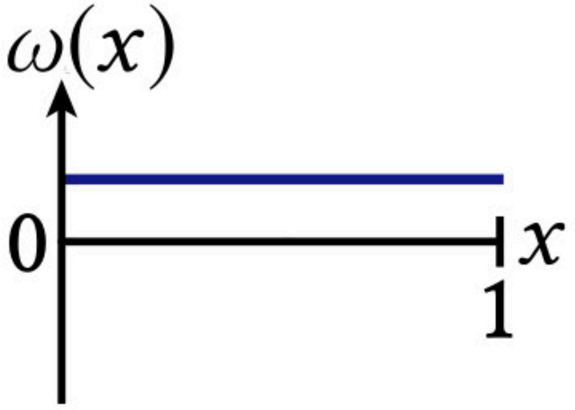
$\omega := \mathcal{D}f$

$+$ jump derivative

Integrating the Darboux derivative

We can only integrate "up to jumps":

Darboux derivative

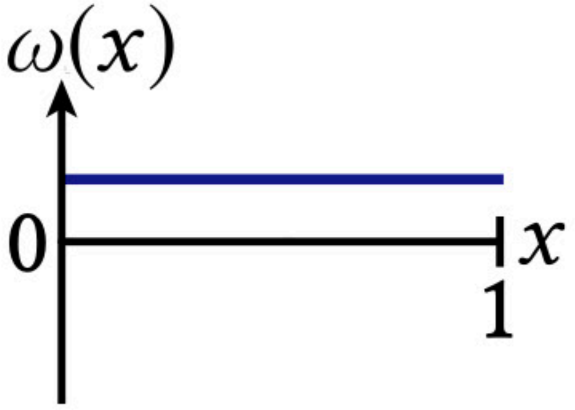


$$\omega := \mathcal{D}f$$

Integrating the Darboux derivative

We can only integrate "up to jumps":

Darboux derivative



integrate

$\mapsto$

$$\omega := \mathcal{D}f$$

Integrating the Darboux derivative

We can only integrate "up to jumps":

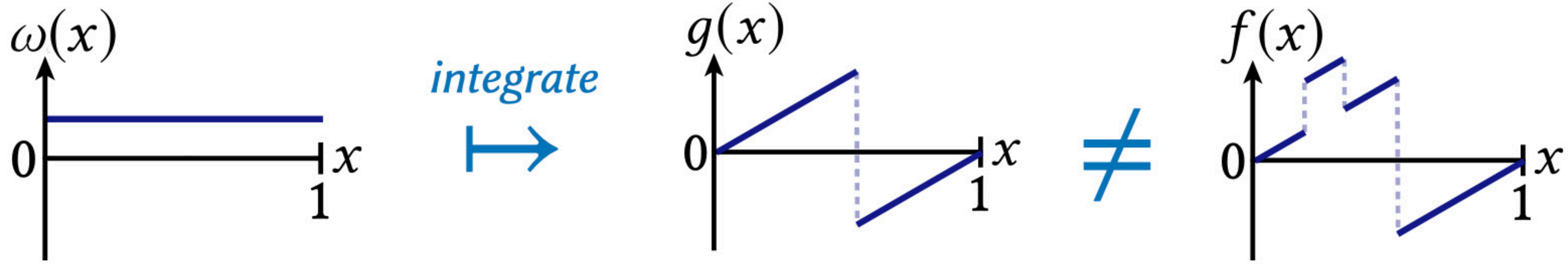
Darboux derivative

$$\omega := \mathcal{D}f$$

Integrating the Darboux derivative

We can only integrate "up to jumps":

Darboux derivative

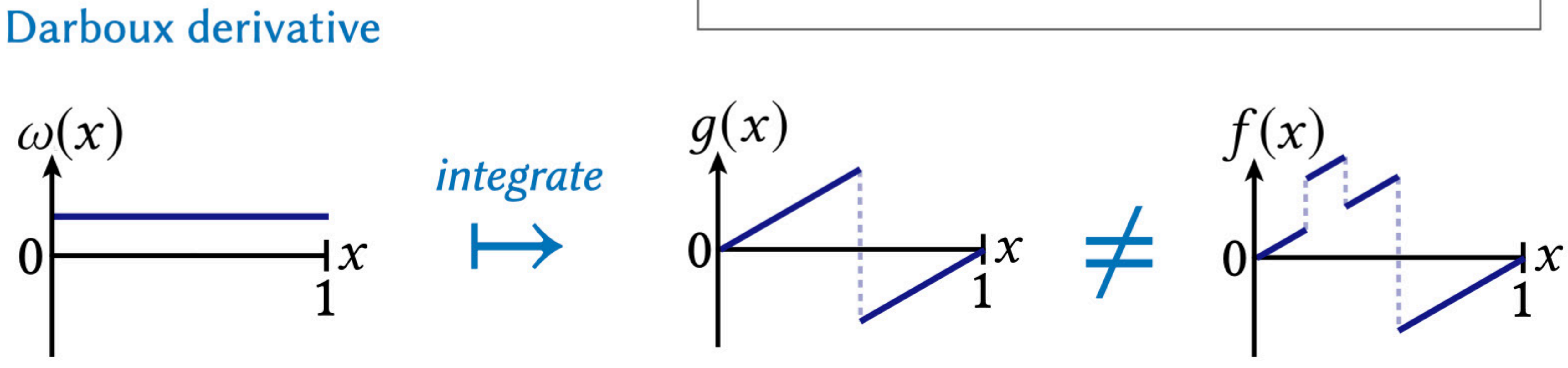


$\omega := \mathcal{D}f$

Integrating the Darboux derivative

We can only integrate "up to jumps":

Moral: To map from derivatives back to curves, we can integrate ω — and choose the jumps!

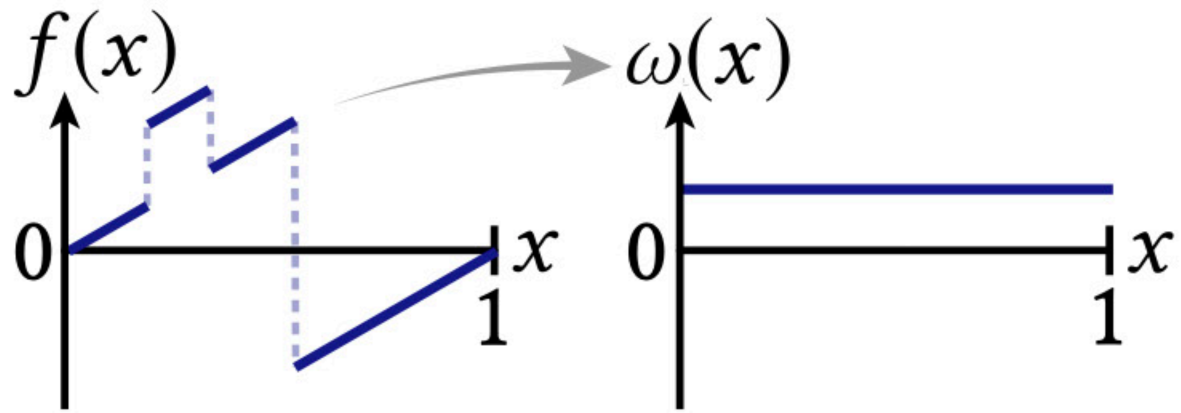


$\omega := \mathcal{D}f$

2D jump harmonic functions — same story

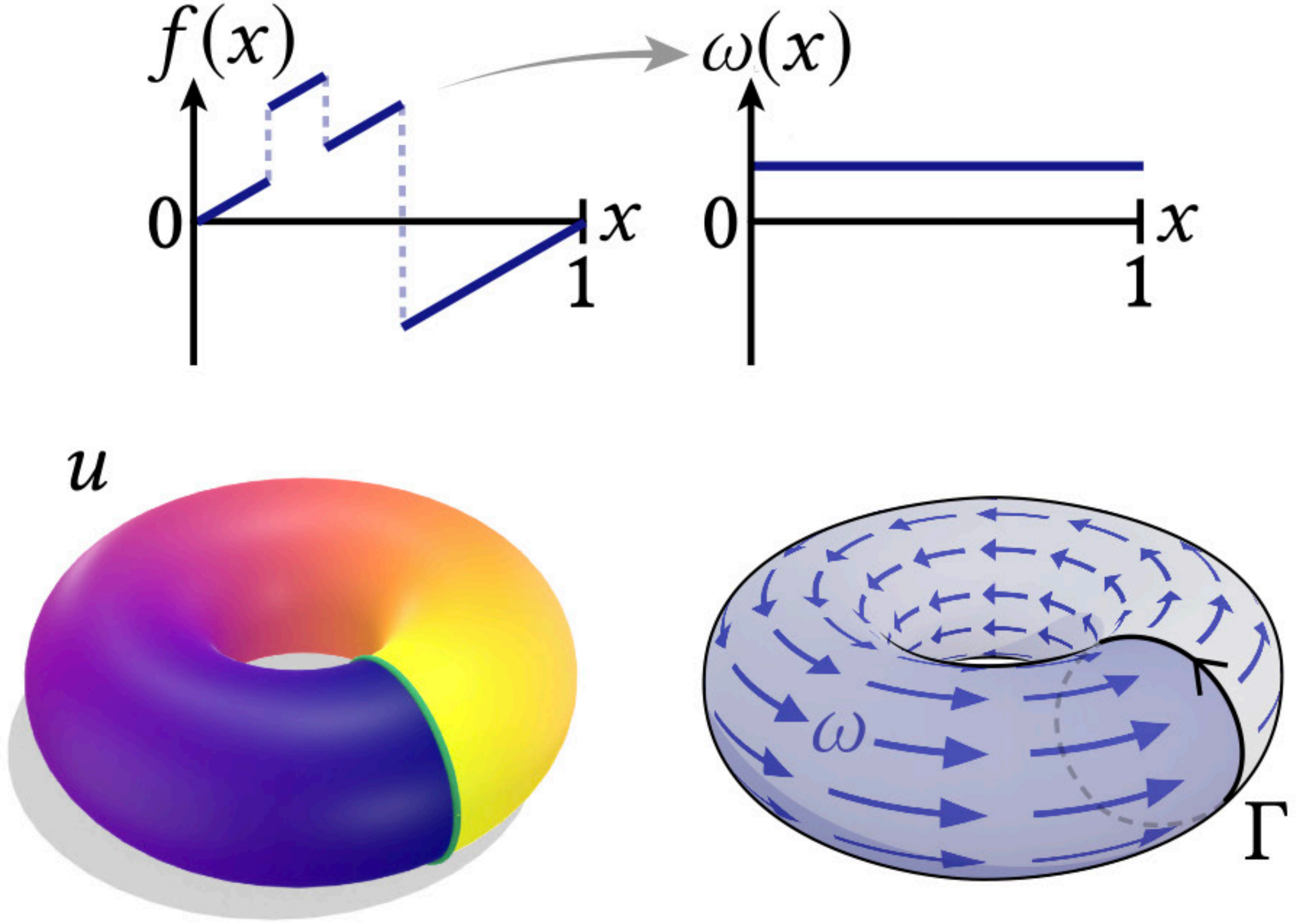
2D jump harmonic functions — same story

For a **jump harmonic** function f, the *Darboux derivative* $\omega := \mathcal{D}f$ "forgets" jumps:



2D jump harmonic functions — same story

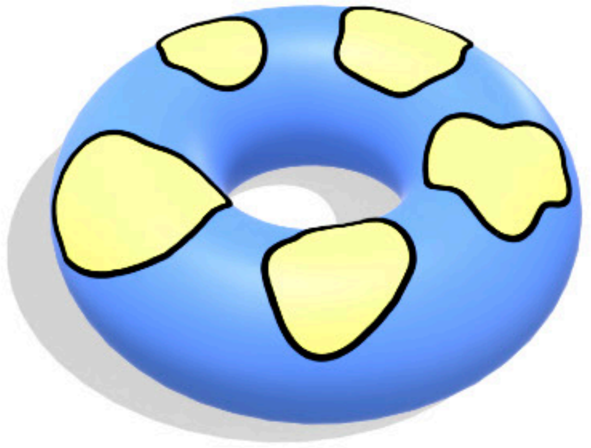
For a **jump harmonic** function *f*, the *Darboux derivative* $\omega := \mathcal{D}f$ "forgets" jumps:



Nonbounding curves $\Longleftrightarrow$ nonzero derivative

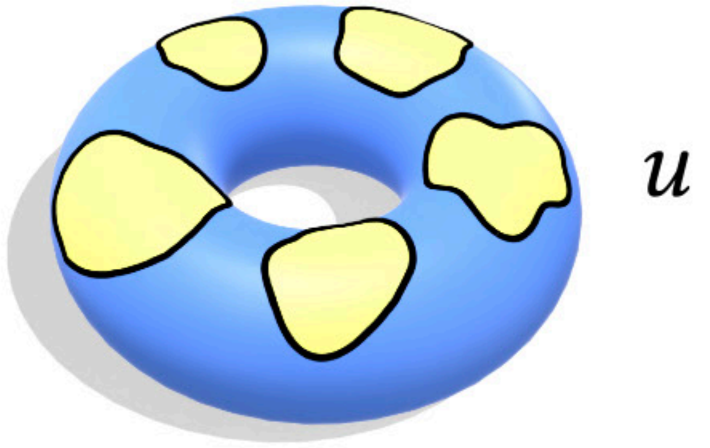
Nonbounding curves $\Longleftrightarrow$ nonzero derivative

If $\omega = 0$, then u is piecewise constant $\Rightarrow$ u is already a valid region labeling.

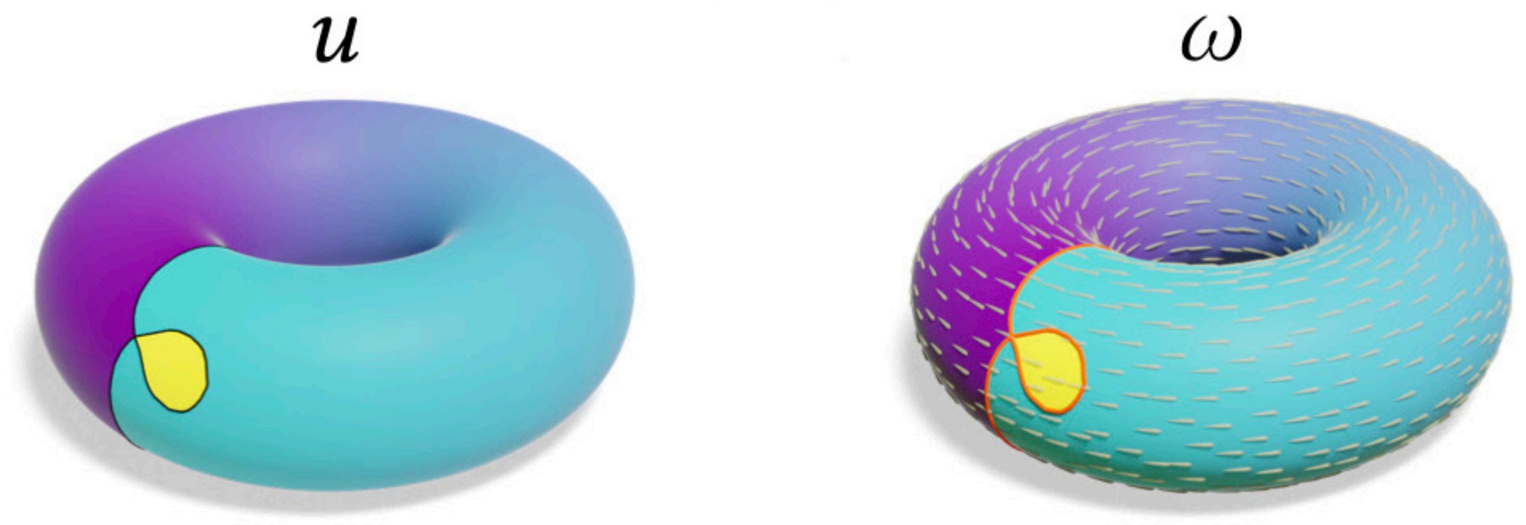


Nonbounding curves $\Longleftrightarrow$ nonzero derivative

If $\omega = 0$, then u is piecewise constant $\Rightarrow$
u is already a valid region labeling.

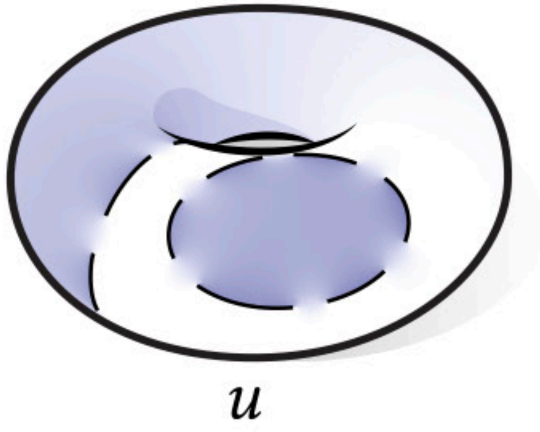


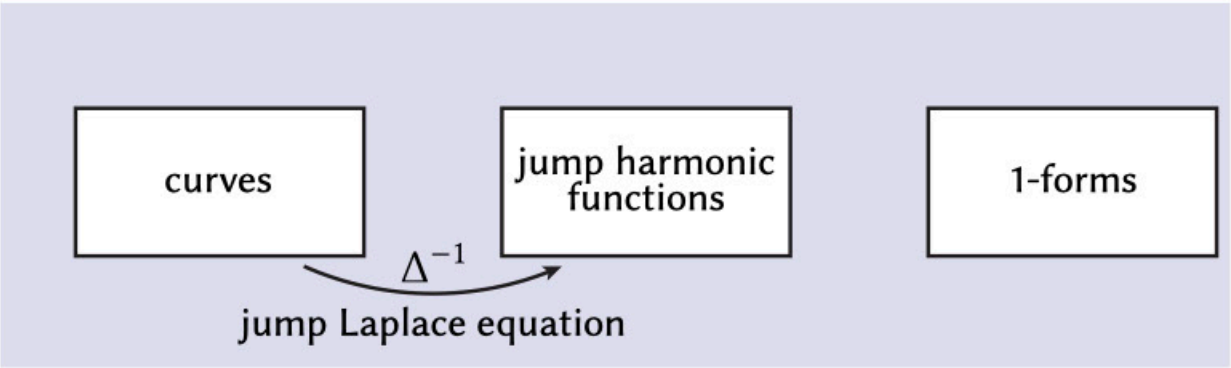
Otherwise, Γ has nonbounding components:



Derivative decomposition

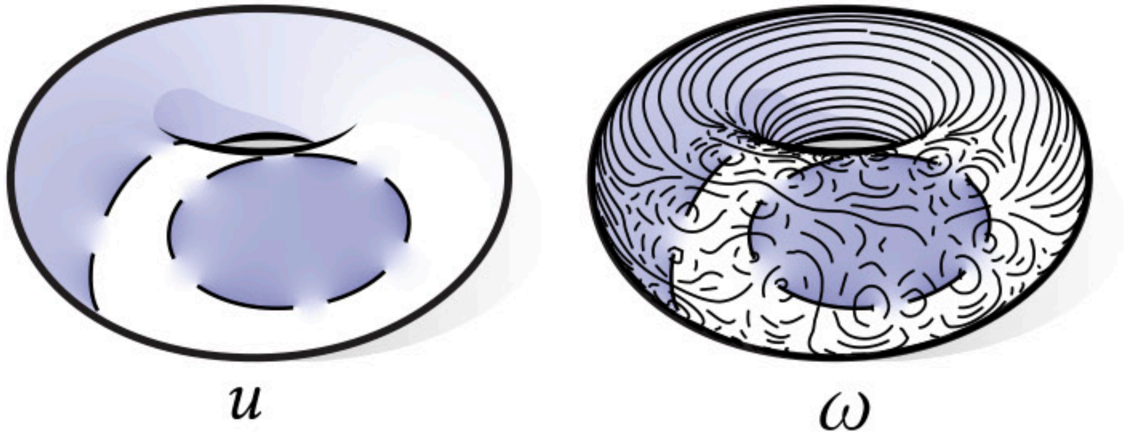
Derivative decomposition

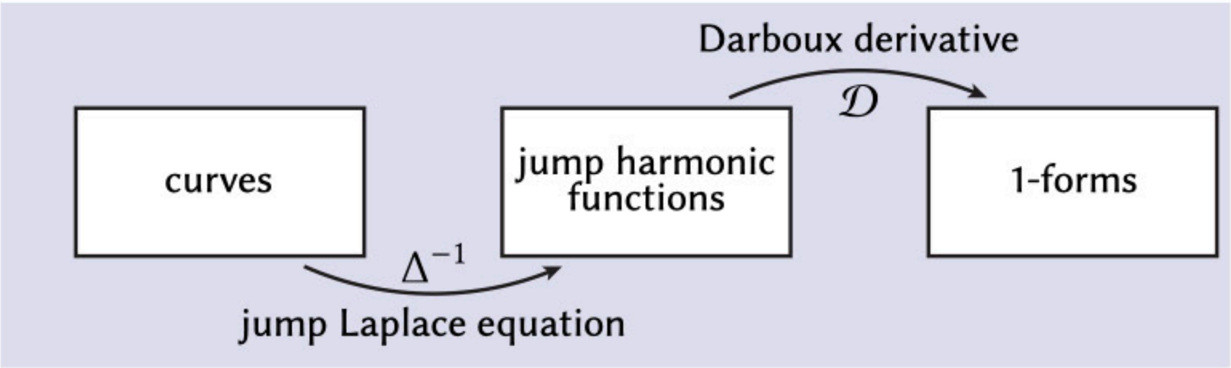




$$\Gamma \xrightarrow{\Delta^{-1}} u$$

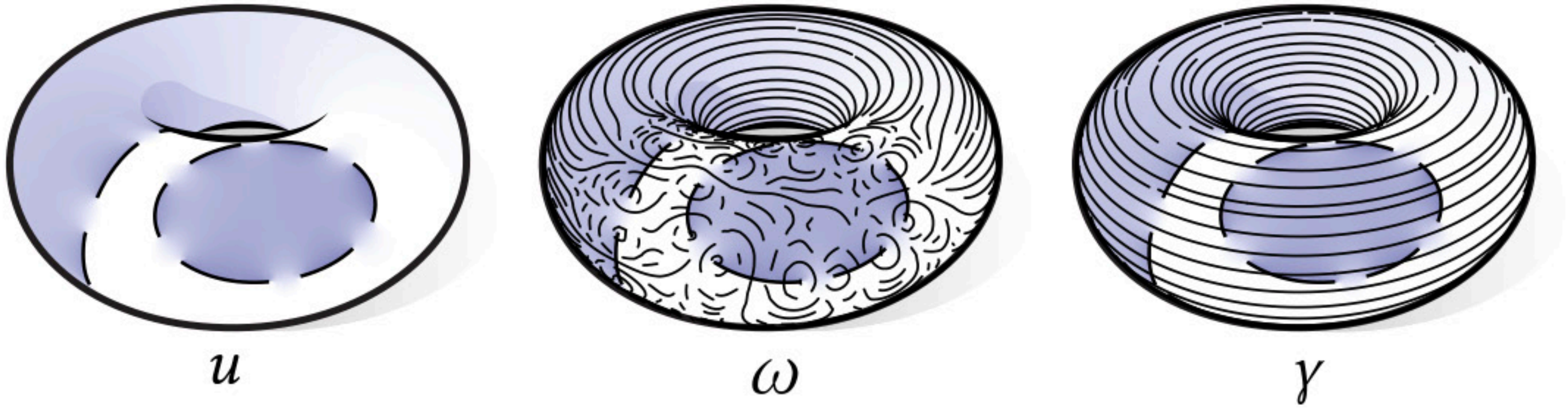
Derivative decomposition





$$\Gamma \xrightarrow{\Delta^{-1}} u \xrightarrow{\mathcal{D}} \omega$$

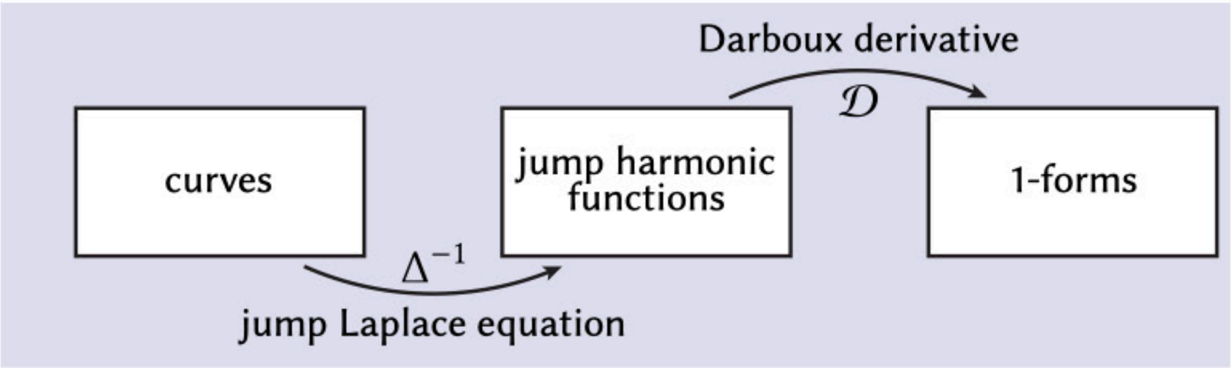
Derivative decomposition



Hodge decomposition:

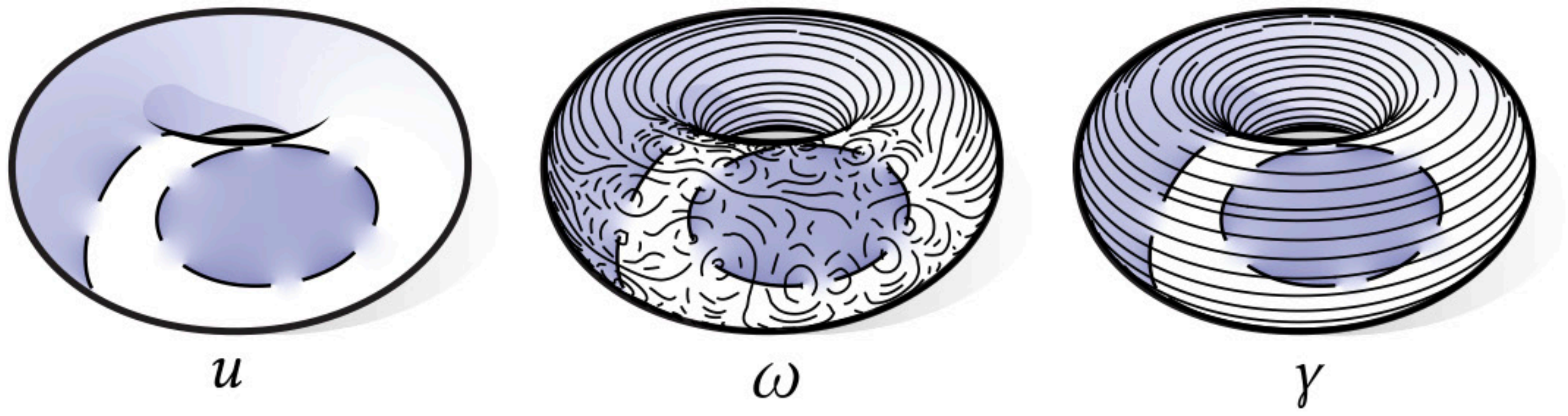
$$\omega = d\alpha + \delta\beta + \gamma$$

γ is a **harmonic 1-form**



$$\Gamma \xrightarrow{\Delta^{-1}} u \xrightarrow{\mathcal{D}} \omega \longrightarrow \gamma$$

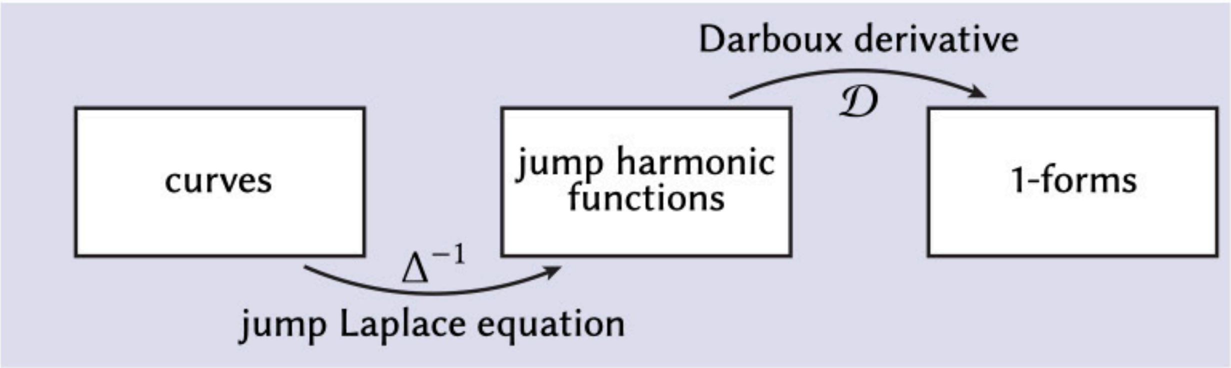
Derivative decomposition

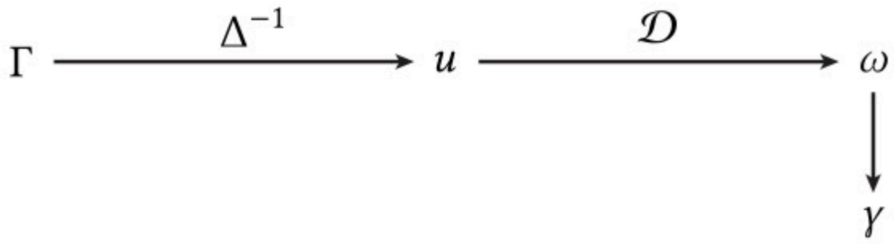


Hodge decomposition:

$$\omega = d\alpha + \delta\beta + \gamma$$

γ is a **harmonic 1-form**

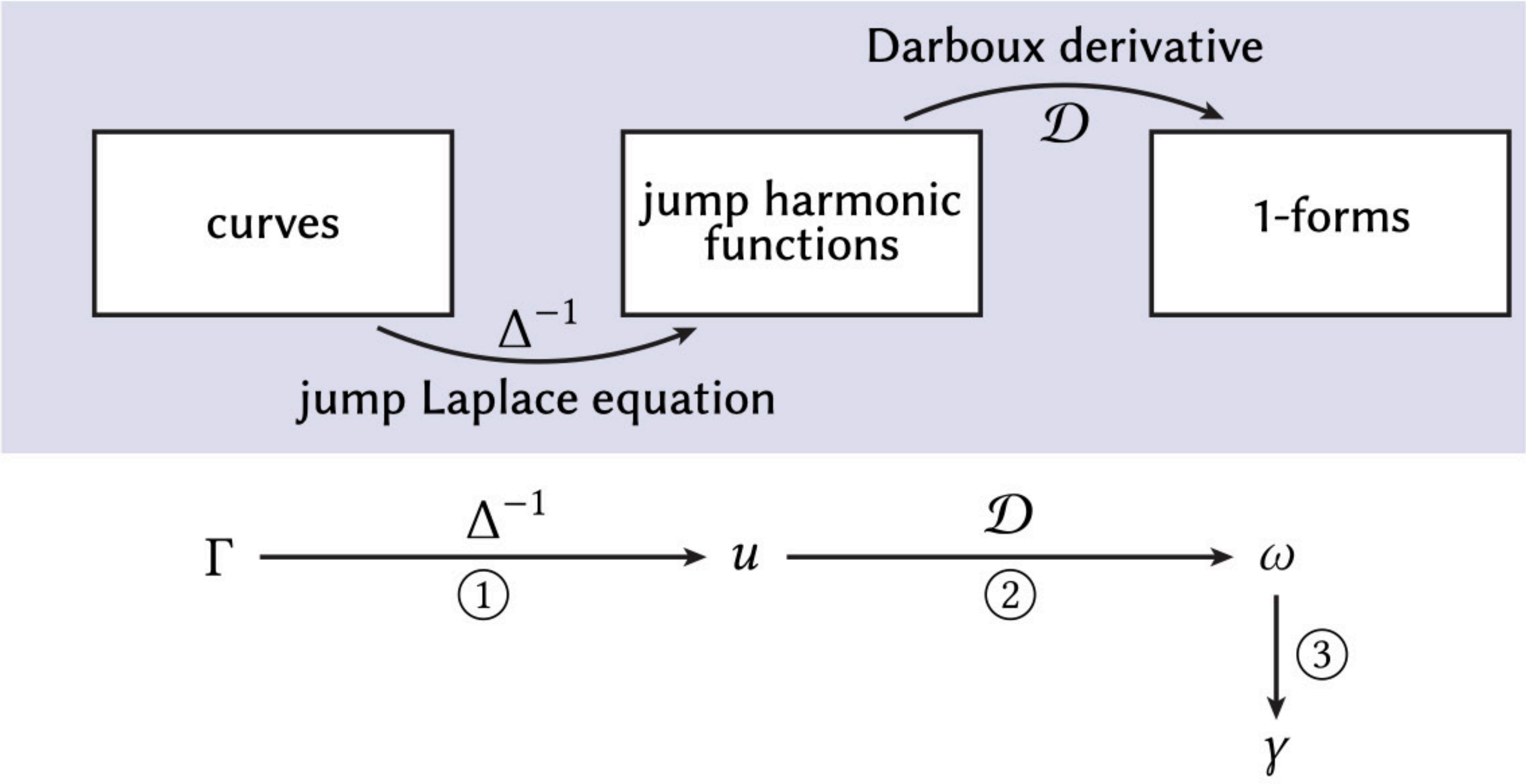




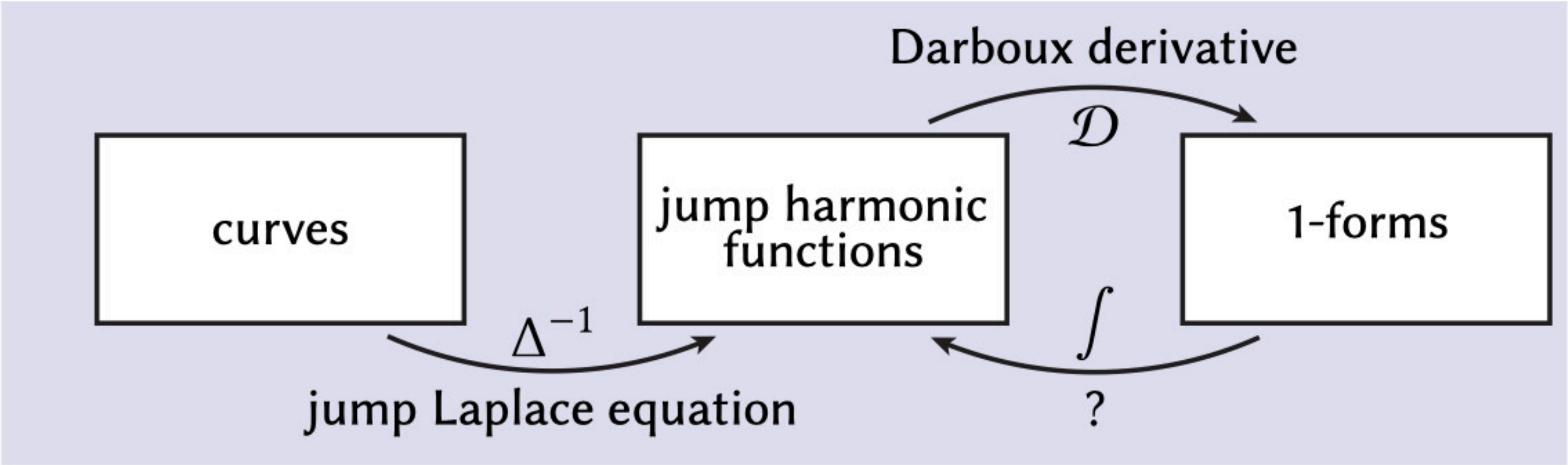
Nonbounding components of Γ are encoded by the harmonic component γ of ω.

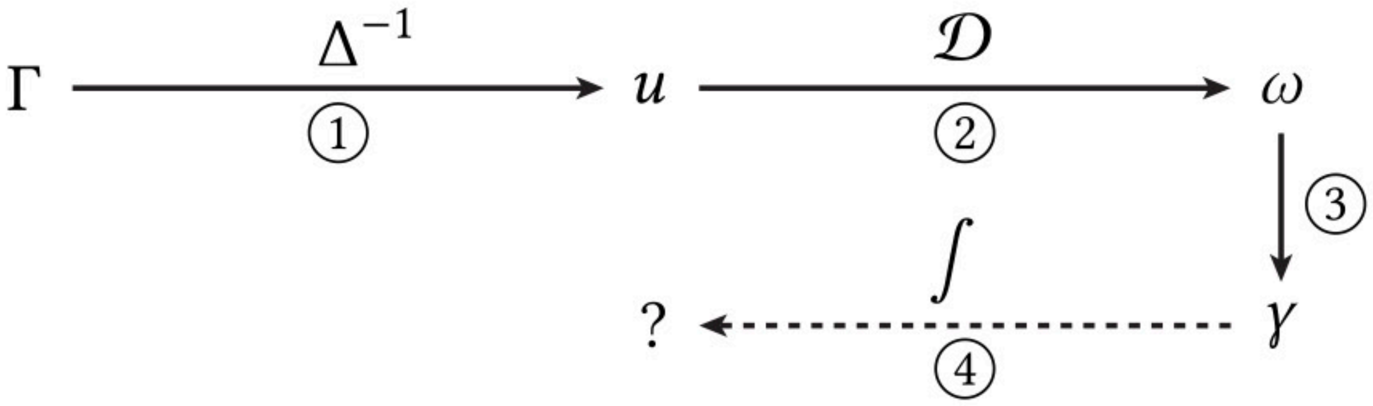
More formally: (Non)bounding components of Γ correspond to 1-forms (non)congruent to zero in the *first cohomology group* $H^1(M) = \ker(d_1) \setminus \operatorname{im}(d_0)$.

Derivative decomposition → curve decomposition



Derivative decomposition → curve decomposition





1-forms → jump harmonic functions

Search for a scalar potential v that could have generated γ.

$$\mathcal{D}v = \gamma$$

1-forms → jump harmonic functions

Search for a scalar potential v that could have generated γ.

$$\mathcal{D}v = \gamma$$

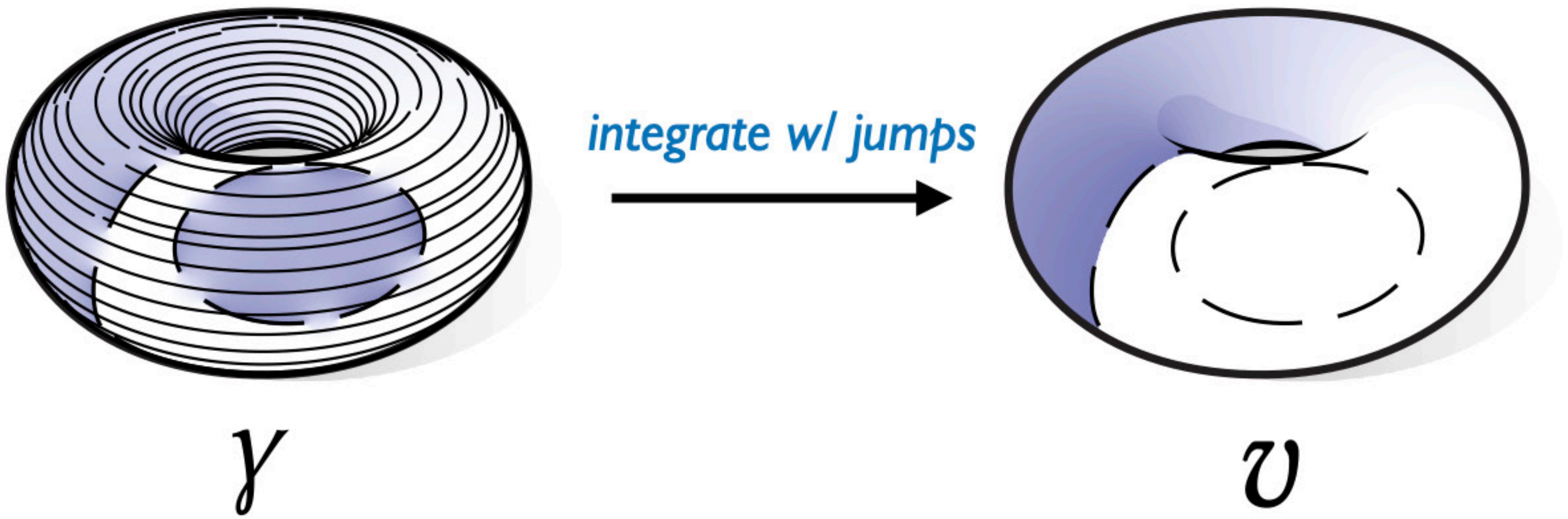
Since γ is harmonic, v must jump somewhere.

1-forms → jump harmonic functions

Search for a scalar potential v that could have generated γ.

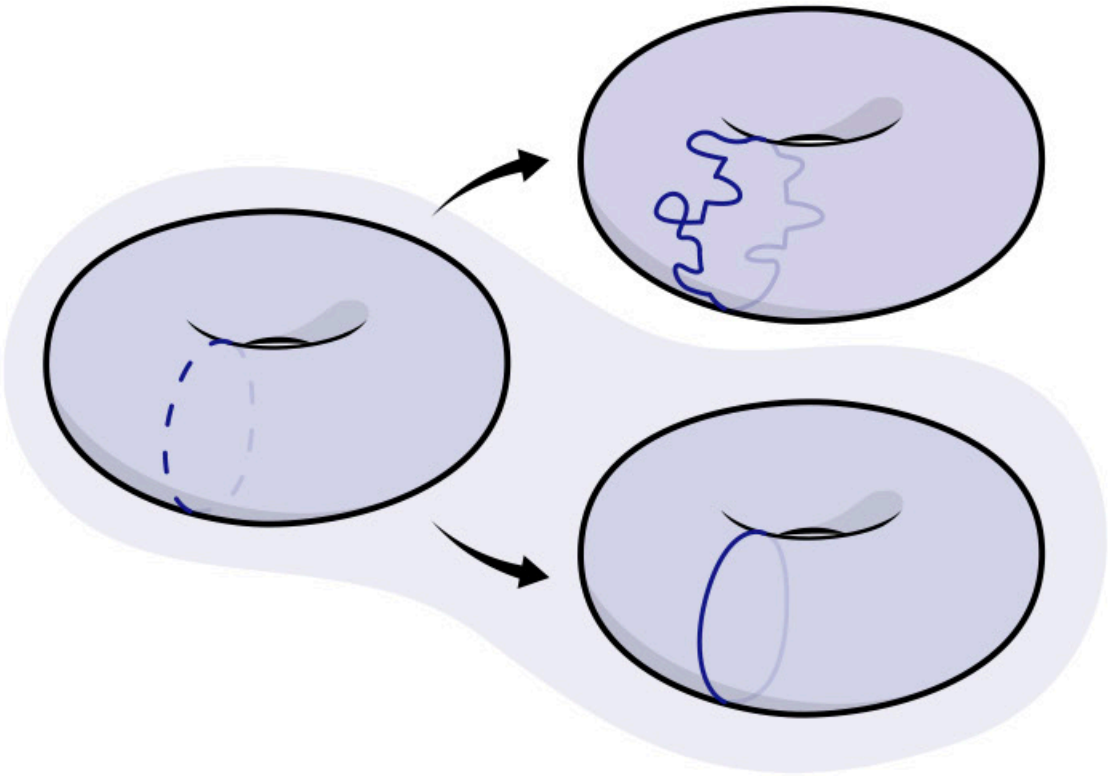
$$\mathcal{D}v = \gamma$$

Since γ is harmonic, v must jump somewhere.

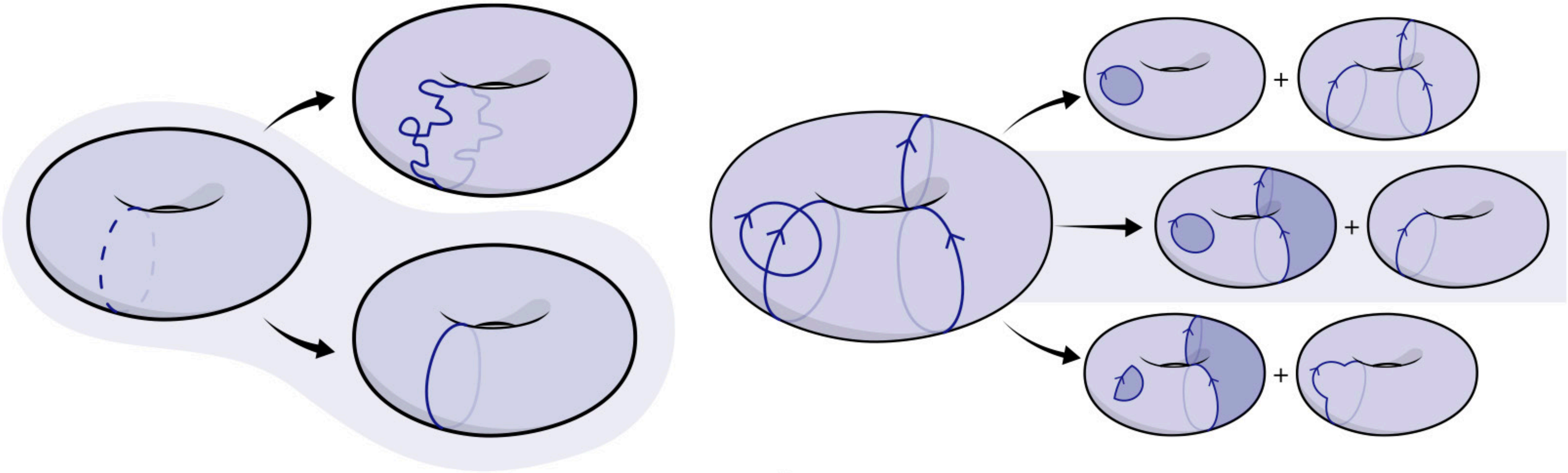


1-forms → jump harmonic functions

1-forms → jump harmonic functions

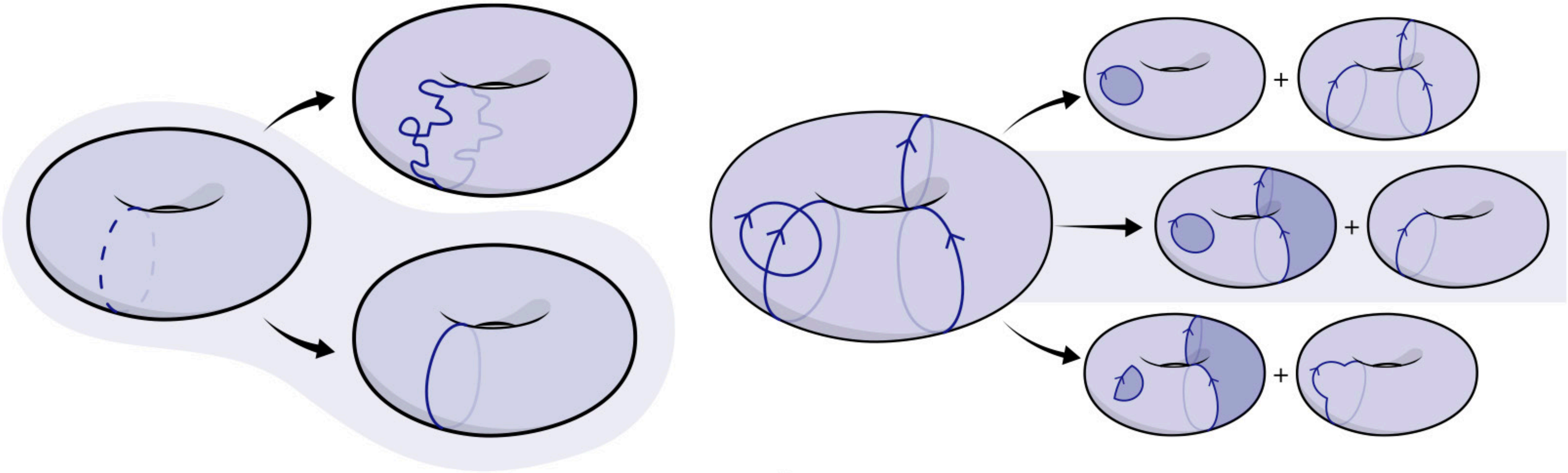


1-forms → jump harmonic functions



1-forms → jump harmonic functions

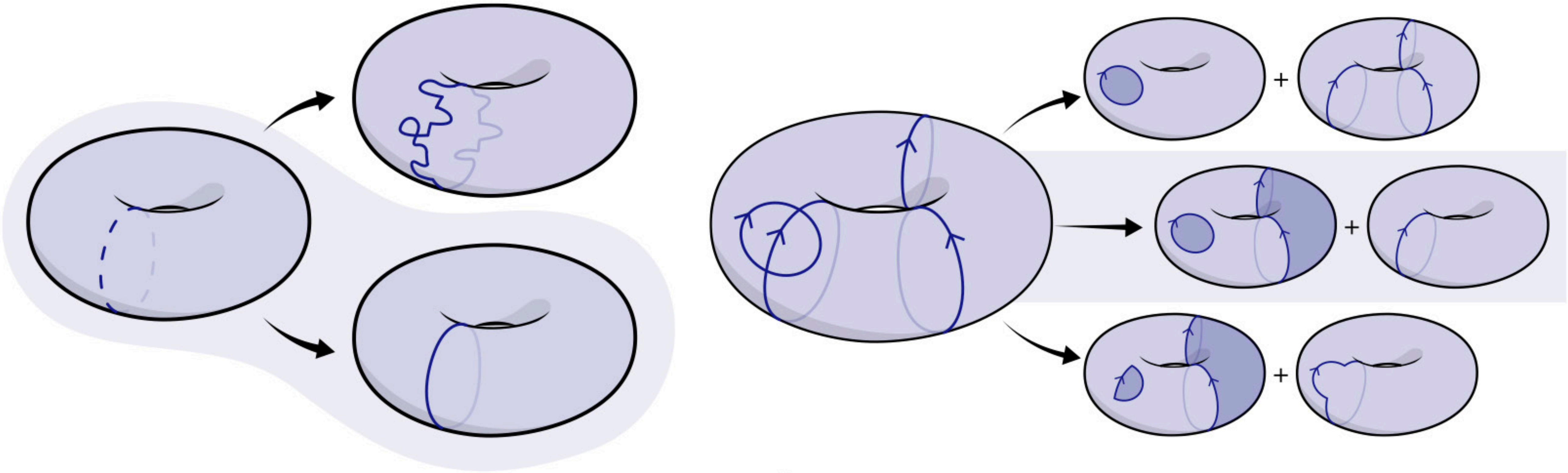
$$\min_{v\colon M\to\mathbb{R}} \int |\text{the jumps not across } \Gamma| + \varepsilon \int |\text{the jumps across } \Gamma|$$



1-forms → jump harmonic functions

penalize jumps

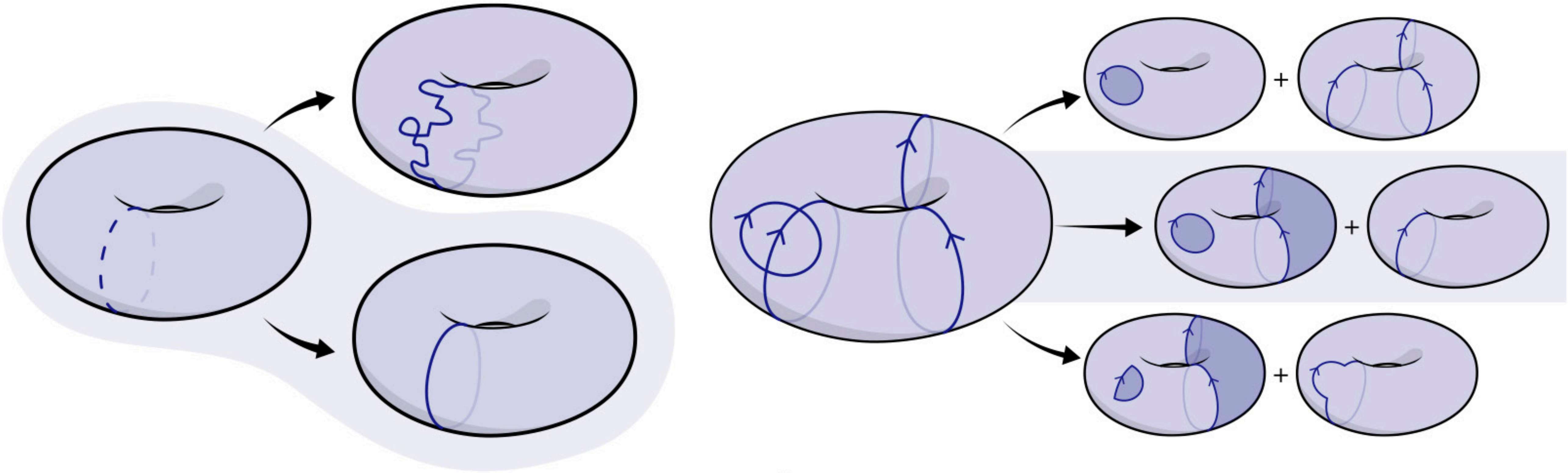
$$\min_{v:\, M \to \mathbb{R}} \boxed{\int |\text{the jumps not across } \Gamma|} + \varepsilon \int |\text{the jumps across } \Gamma|$$



1-forms → jump harmonic functions

penalize jumps | *smaller penalty across Γ*

$$\min_{v:\, M\to\mathbb{R}} \boxed{\int |\text{the jumps not across } \Gamma|} + \boxed{\varepsilon \int |\text{the jumps across } \Gamma|}$$



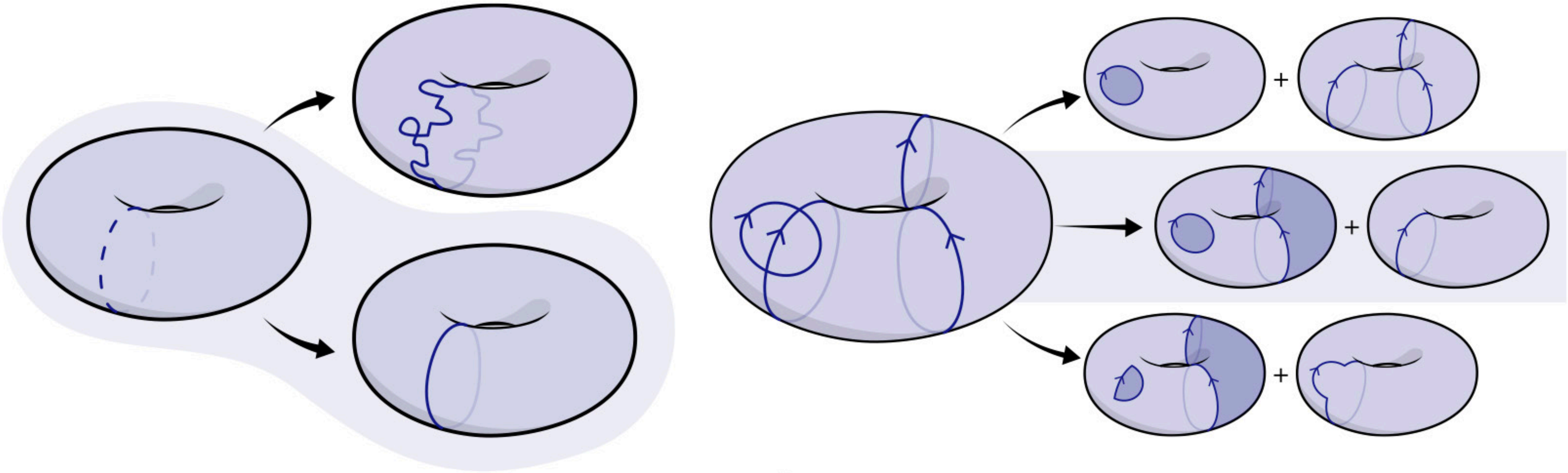
1-forms → jump harmonic functions

penalize jumps

smaller penalty across Γ

$$\min_{v:\, M\to\mathbb{R}} \int |\text{the jumps not across } \Gamma| + \varepsilon \int |\text{the jumps across } \Gamma|$$

concentrate jumps across Γ



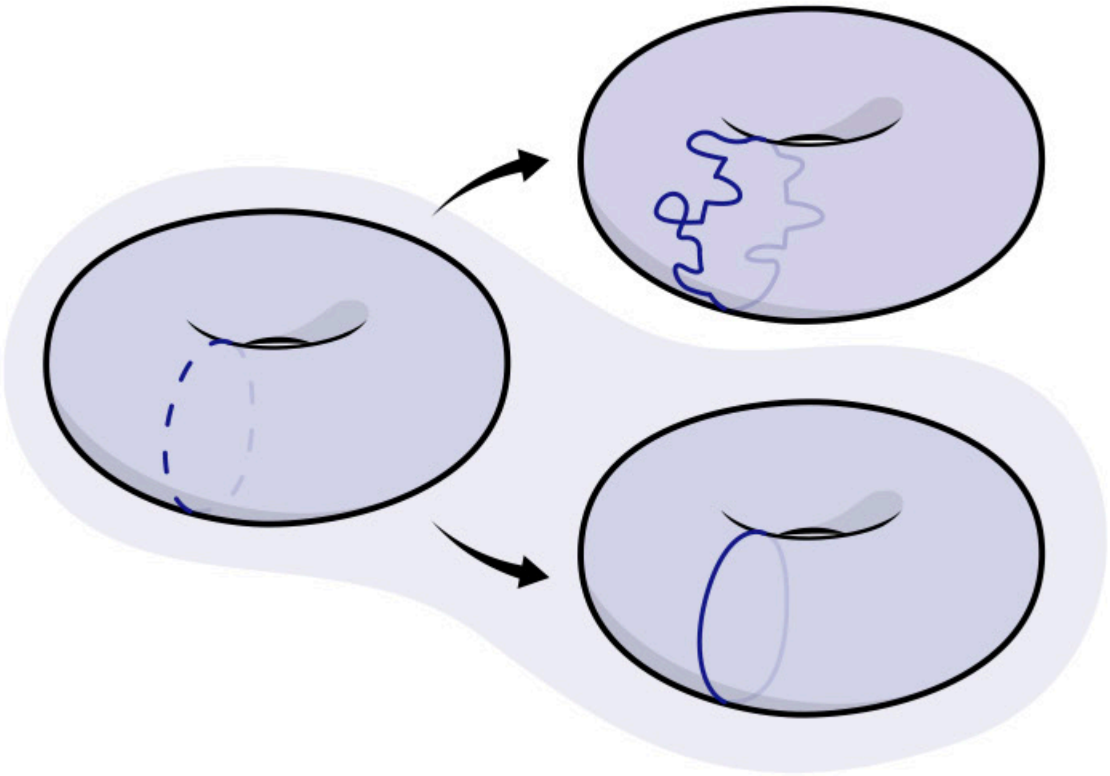
1-forms → jump harmonic functions

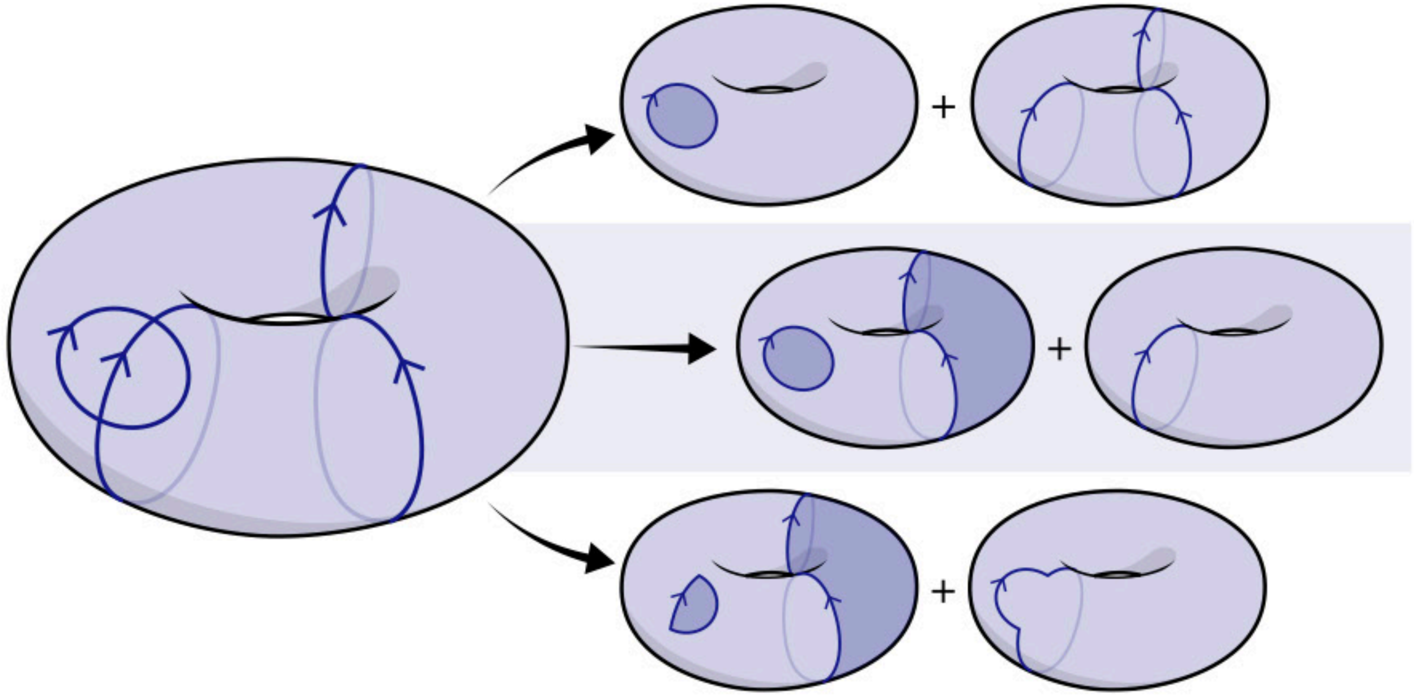
penalize jumps *smaller penalty across Γ*

$$\min_{v:\, M\to\mathbb{R}} \int |\text{the jumps not across } \Gamma| + \varepsilon \int |\text{the jumps across } \Gamma|$$

concentrate jumps across Γ

subject to $\mathcal{D}v = \gamma$ *(co)homology constraint*





1-forms → jump harmonic functions

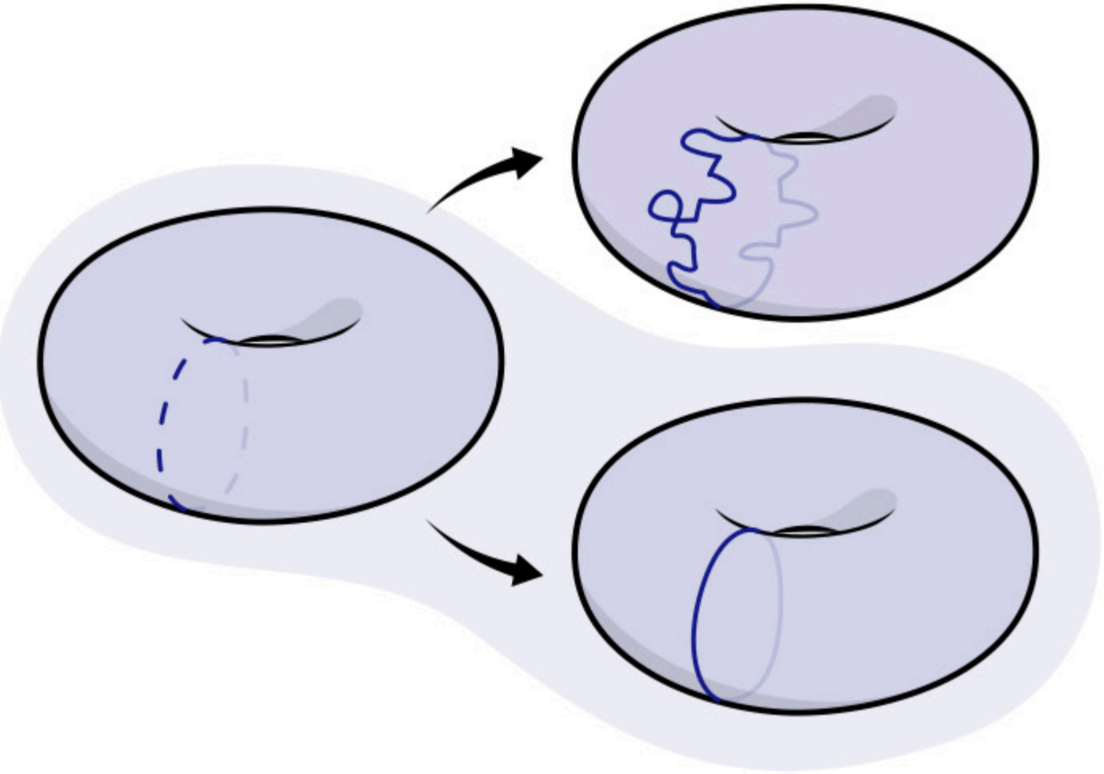
penalize jumps — *smaller penalty across Γ*

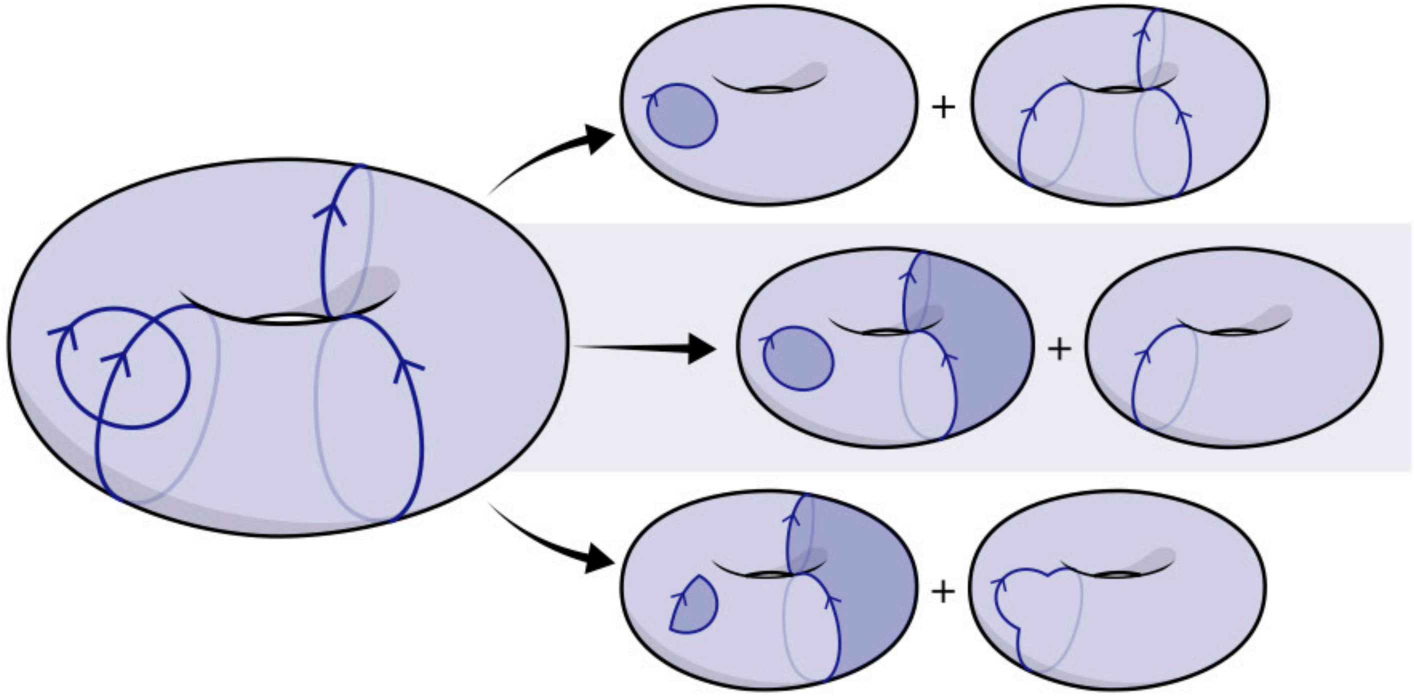
pick shortest completion in homology class

$$\min_{v:\, M\to\mathbb{R}} \int |\text{the jumps not across }\Gamma| + \varepsilon \int |\text{the jumps across }\Gamma|$$

concentrate jumps across Γ

subject to $\mathcal{D}v = \gamma$ *(co)homology constraint*





1-forms → jump harmonic functions

penalize jumps

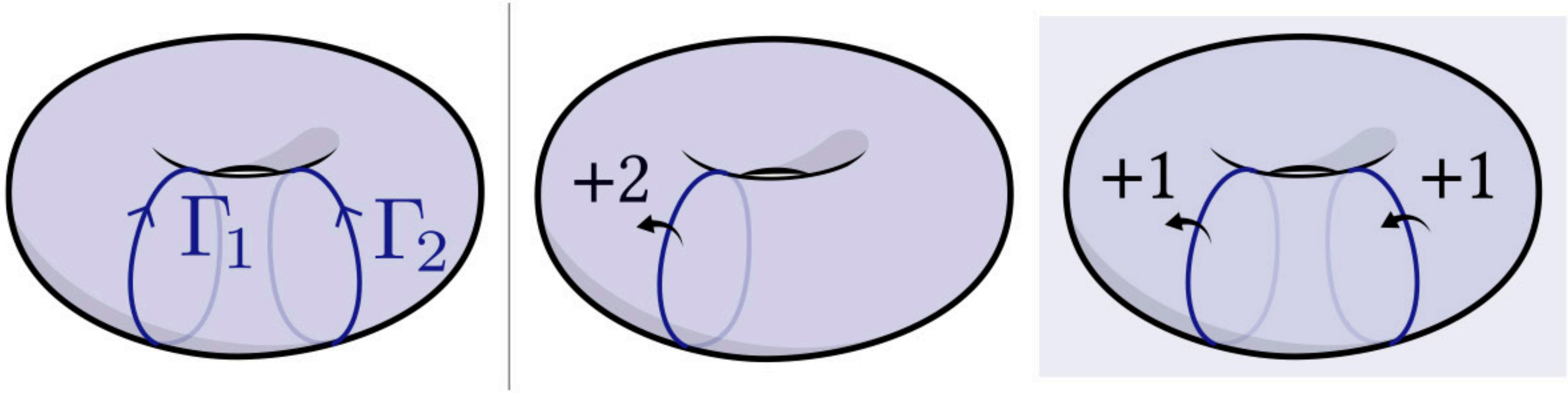
smaller penalty across Γ

pick shortest completion in homology class

$$\min_{v:\, M \to \mathbb{R}} \boxed{\int |\text{the jumps not across } \Gamma|} + \boxed{\varepsilon \int |\text{the jumps across } \Gamma|}$$

concentrate jumps across Γ

subject to $\mathcal{D}v = \gamma$ *(co)homology constraint*



input v

1-forms → jump harmonic functions

pick shortest completion in homology class

penalize jumps — *smaller penalty across Γ*

$$\min_{v\colon M\to\mathbb{R}} \int |\text{the jumps not across } \Gamma| + \varepsilon \int |\text{the jumps across } \Gamma|$$

concentrate jumps across Γ

$$\text{subject to} \qquad \mathcal{D}v = \gamma$$

(co)homology constraint

$$0 \le \frac{v^+ - v^-}{u^+ - u^-} \le 1 \quad \text{on } \Gamma$$

no extra loops

1-forms → jump harmonic functions

penalize jumps *smaller penalty across Γ*

pick shortest completion in homology class

$$\min_{v:\, M\to\mathbb{R}} \int |\text{the jumps not across } \Gamma| + \varepsilon \int |\text{the jumps across } \Gamma|$$

concentrate jumps across Γ

$$\text{subject to} \qquad \mathcal{D}v = \gamma$$

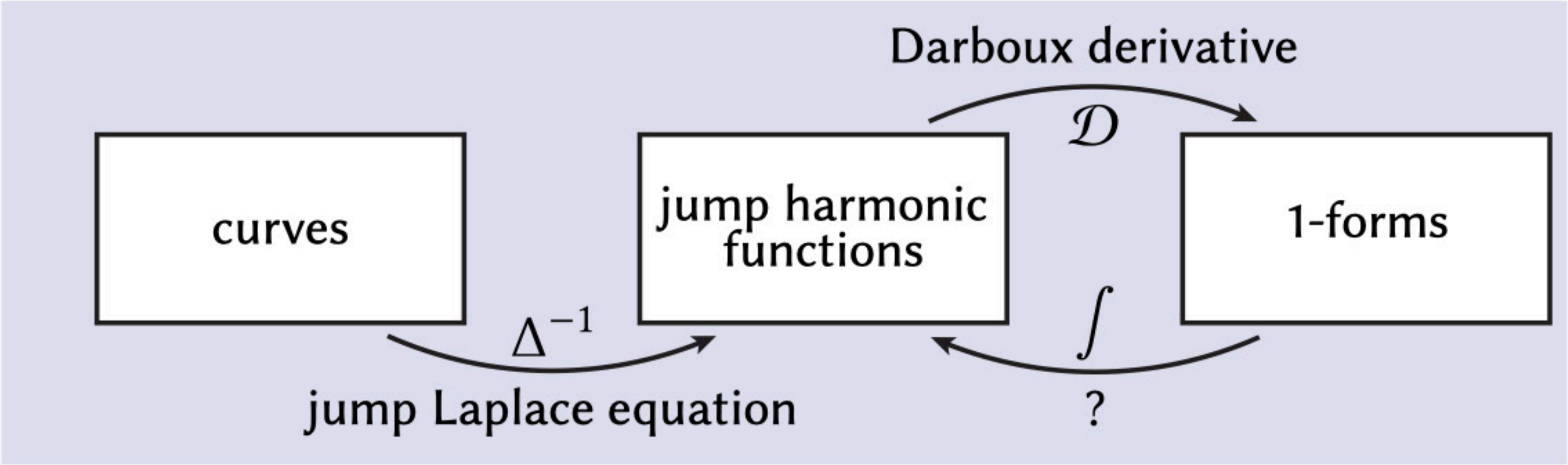
(co)homology constraint

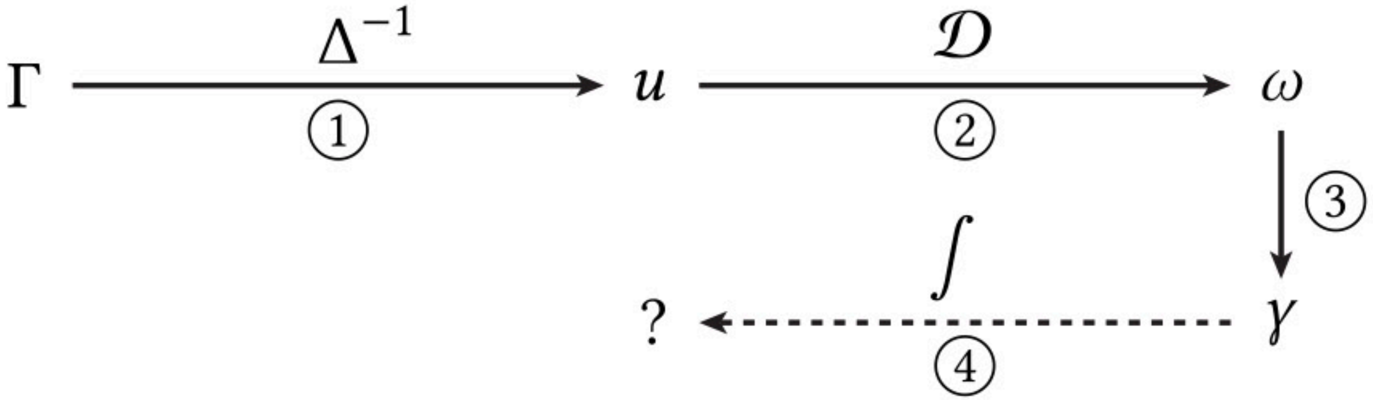
$$0 \le \frac{v^+ - v^-}{u^+ - u^-} \le 1 \quad \text{on } \Gamma$$

no extra loops

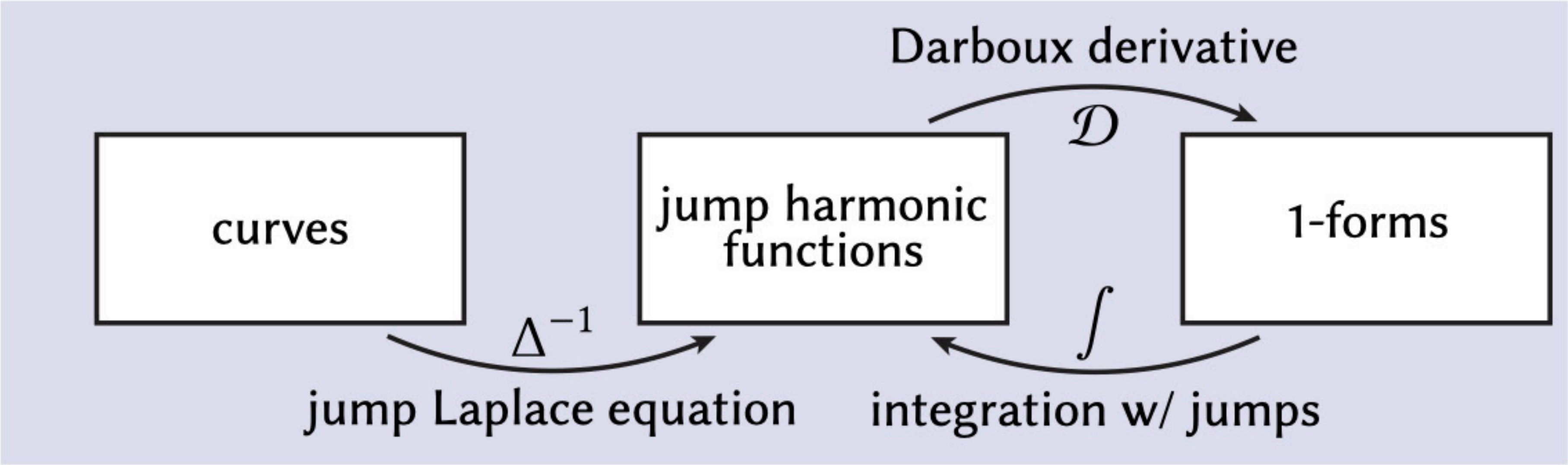
"residual function"

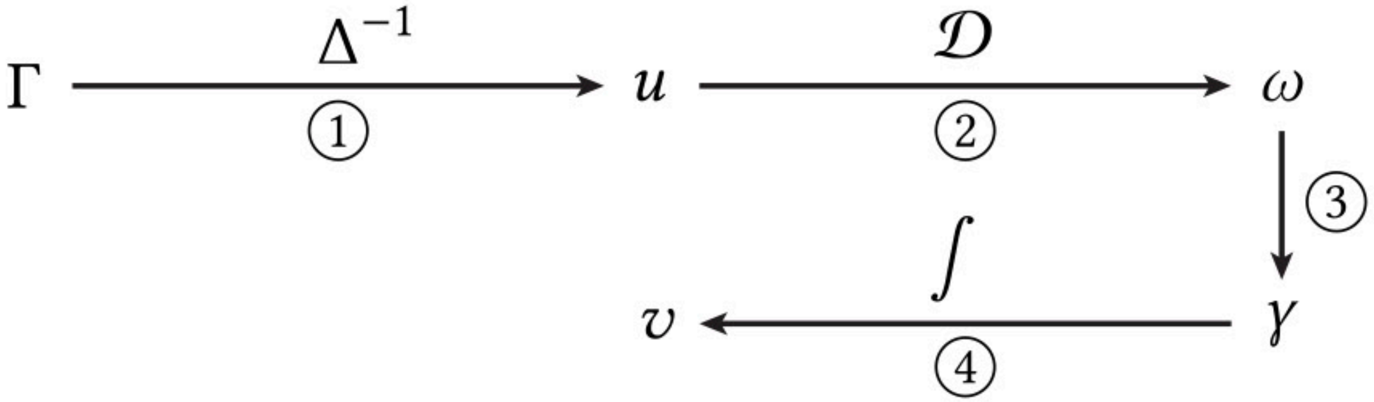
1-forms → jump harmonic functions



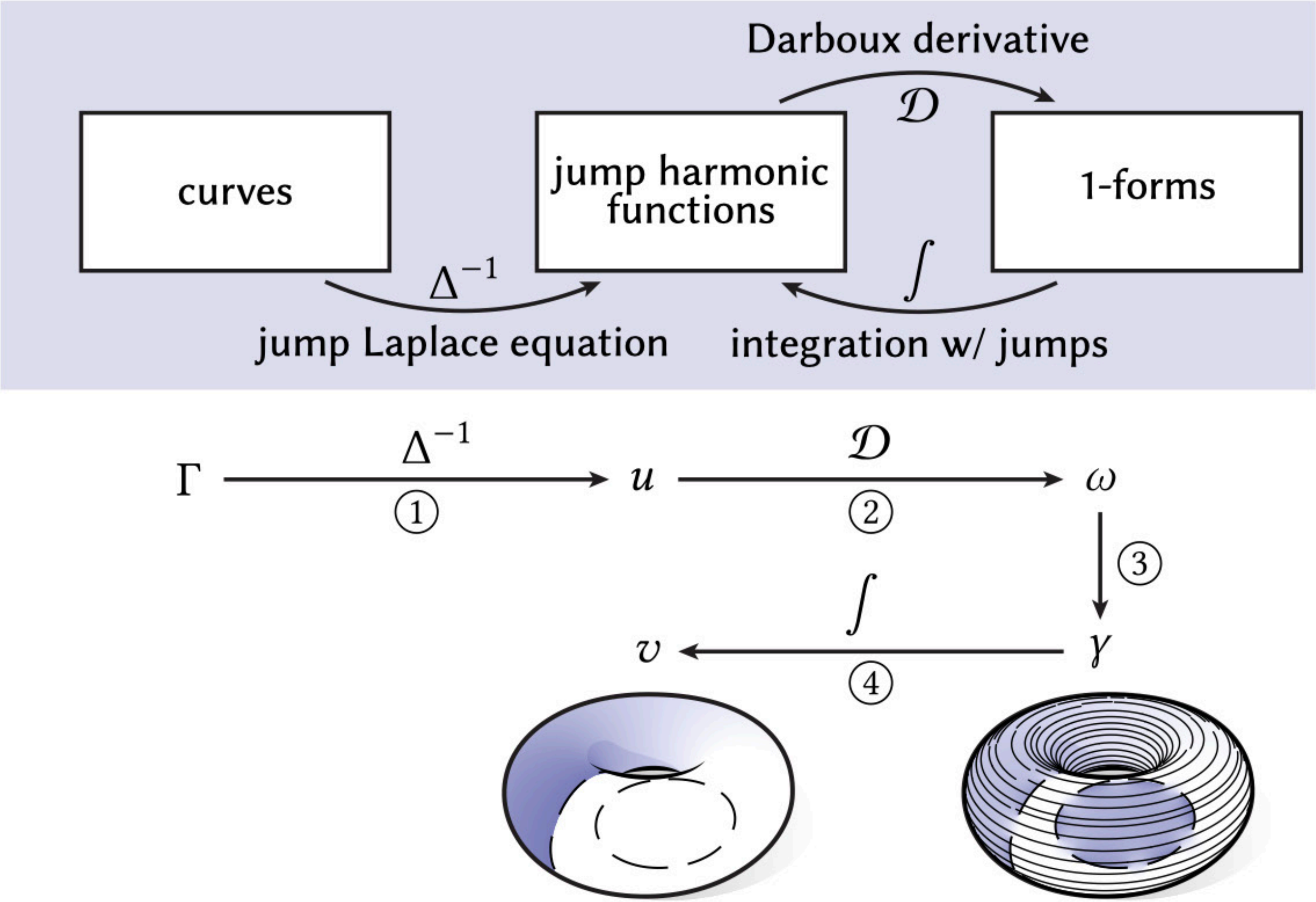


1-forms → jump harmonic functions

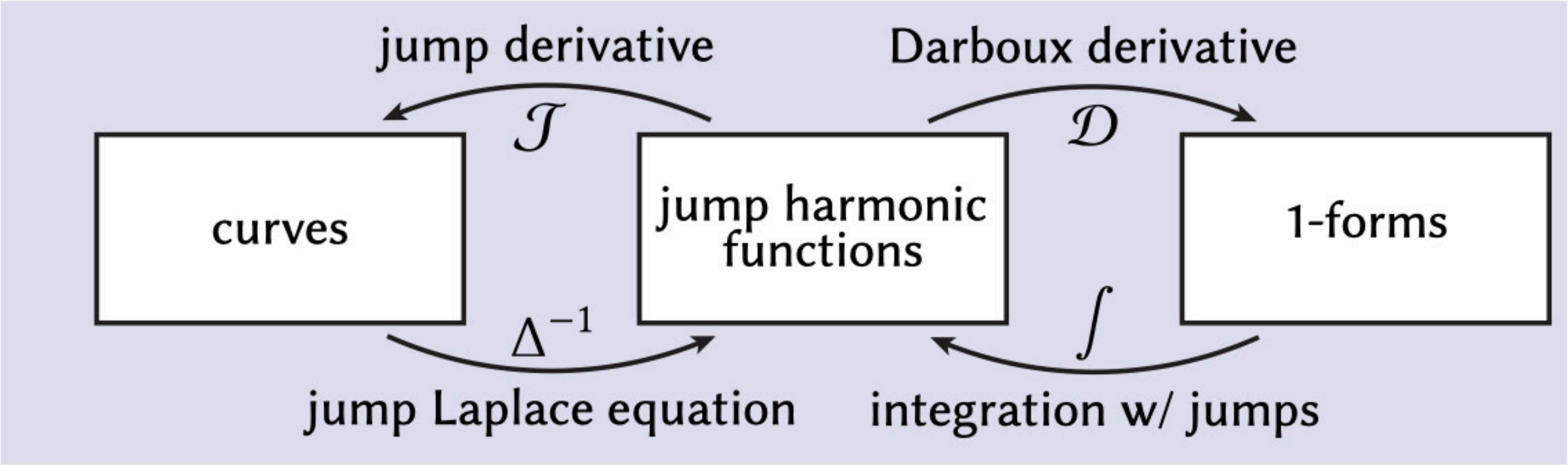


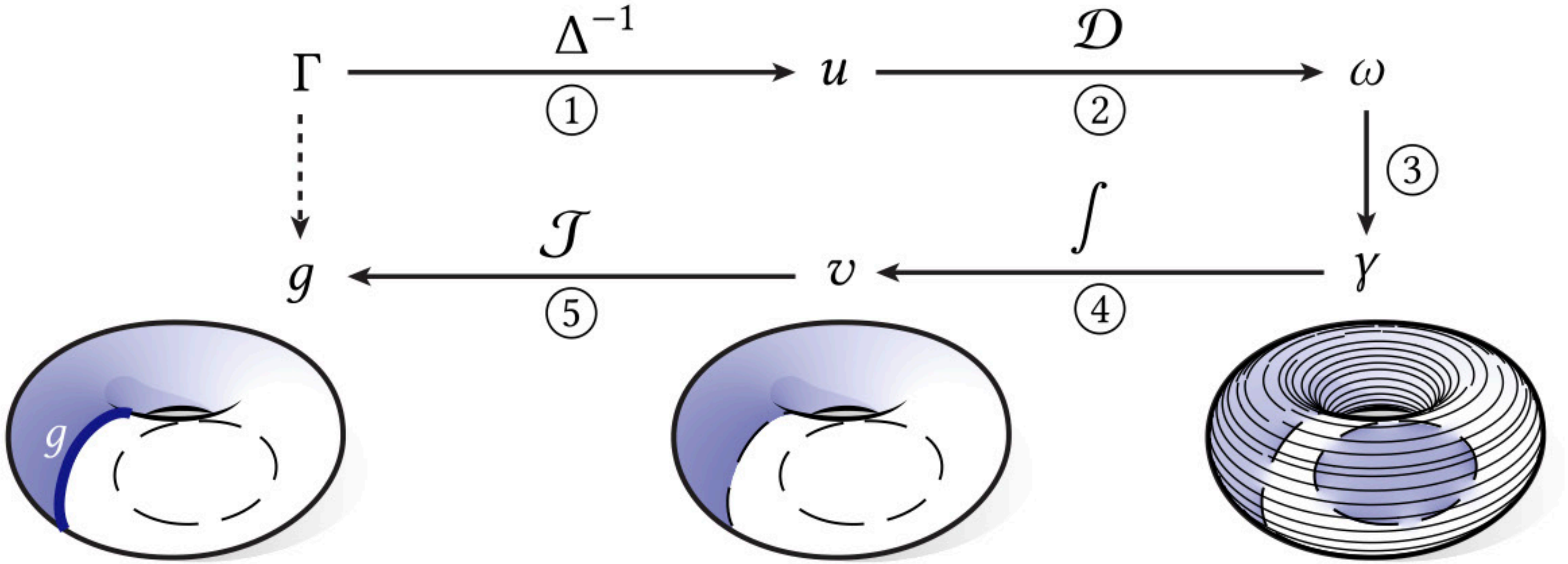


1-forms → jump harmonic functions



Jump harmonic function → curve decomposition

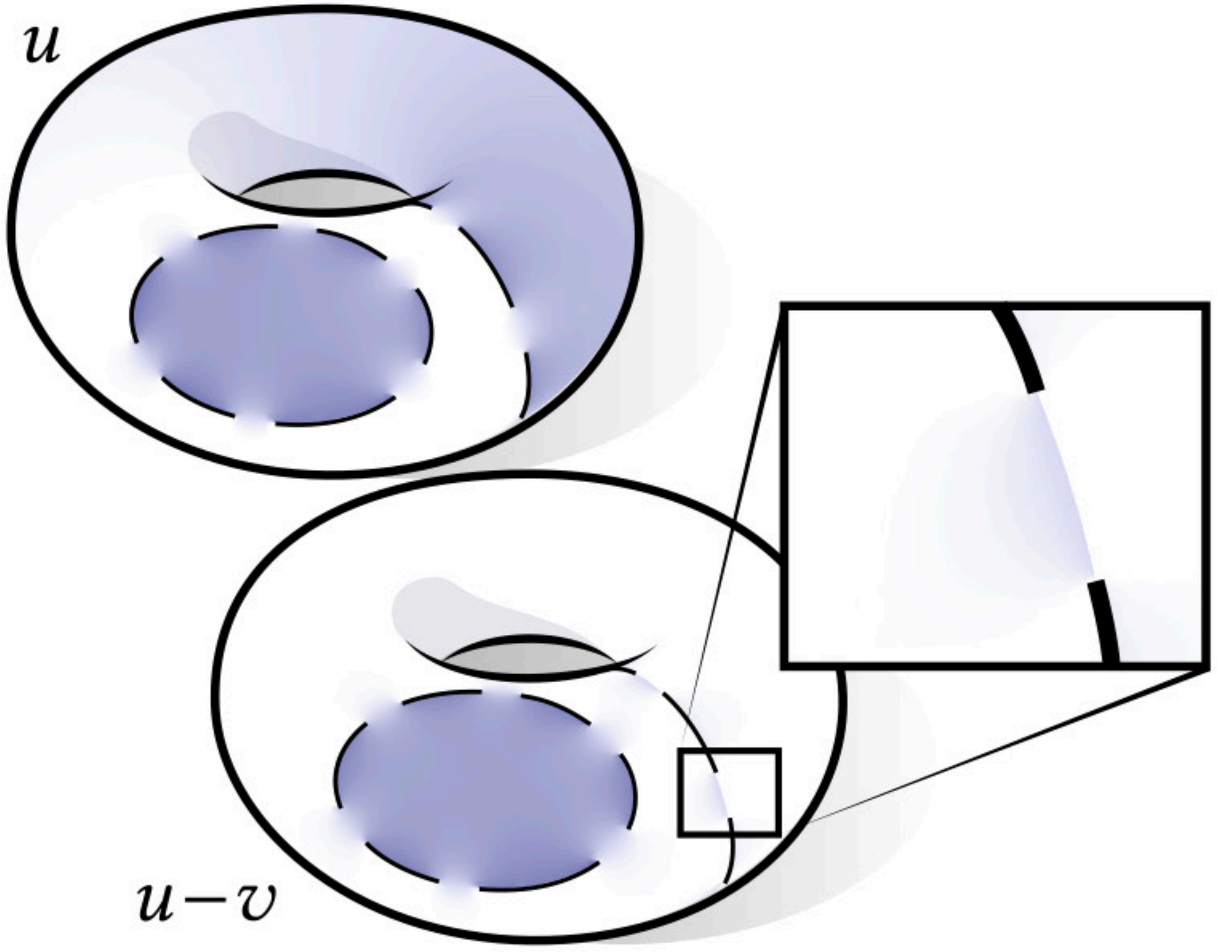




Winding number function

Winding number function

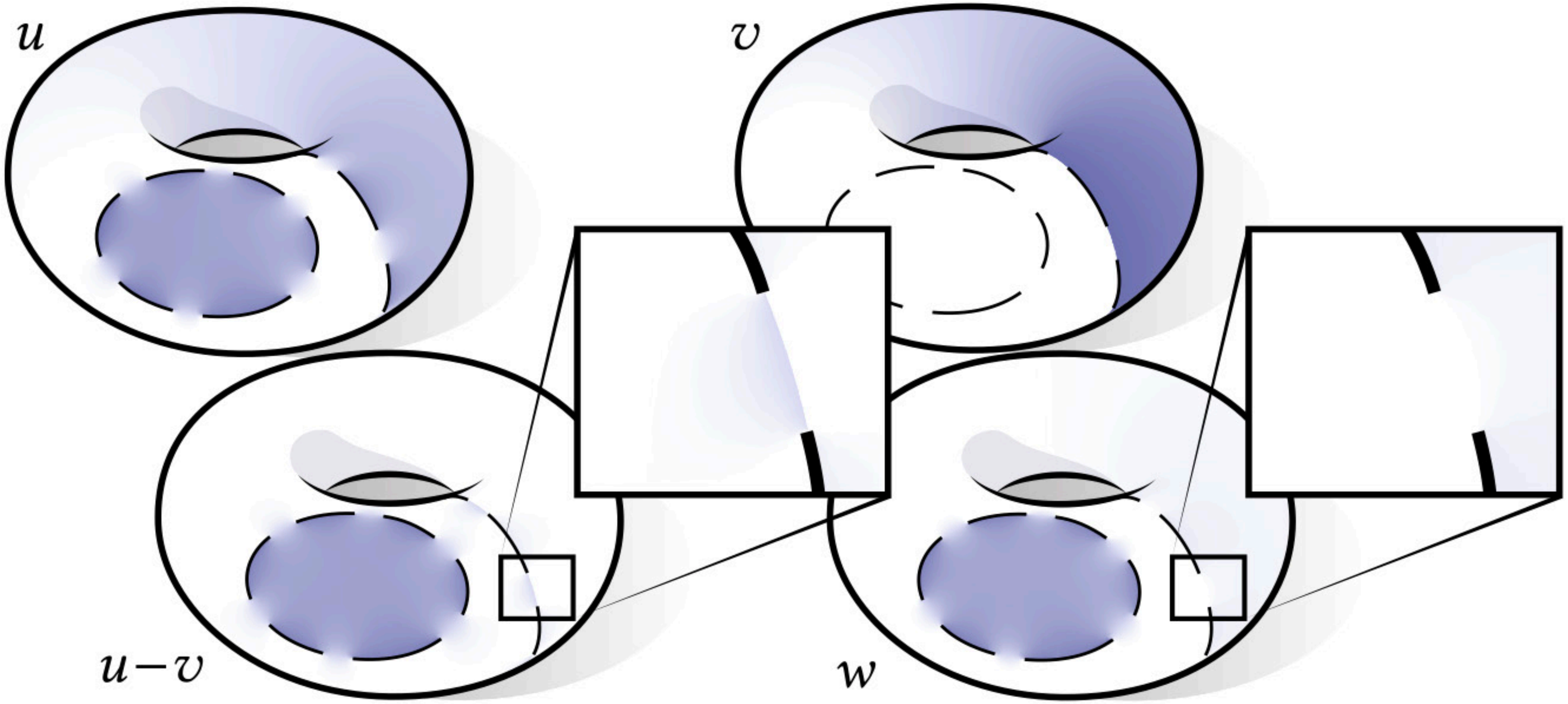
Simply subtracting the residual function
yields extraneous discontinuities.



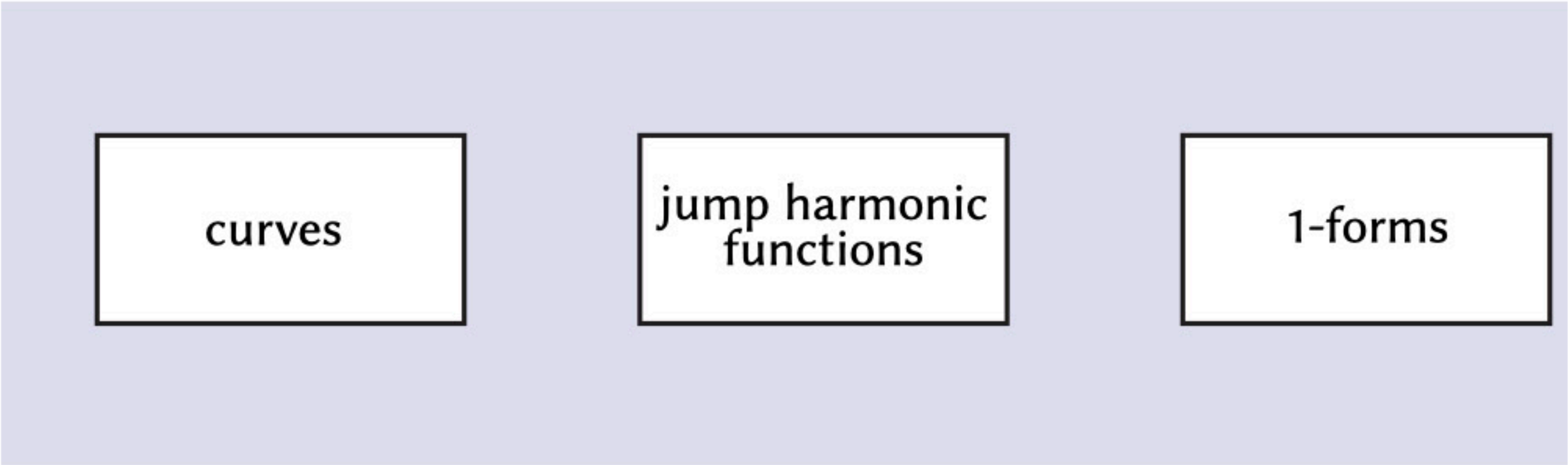
Winding number function

Simply subtracting the residual function yields extraneous discontinuities.

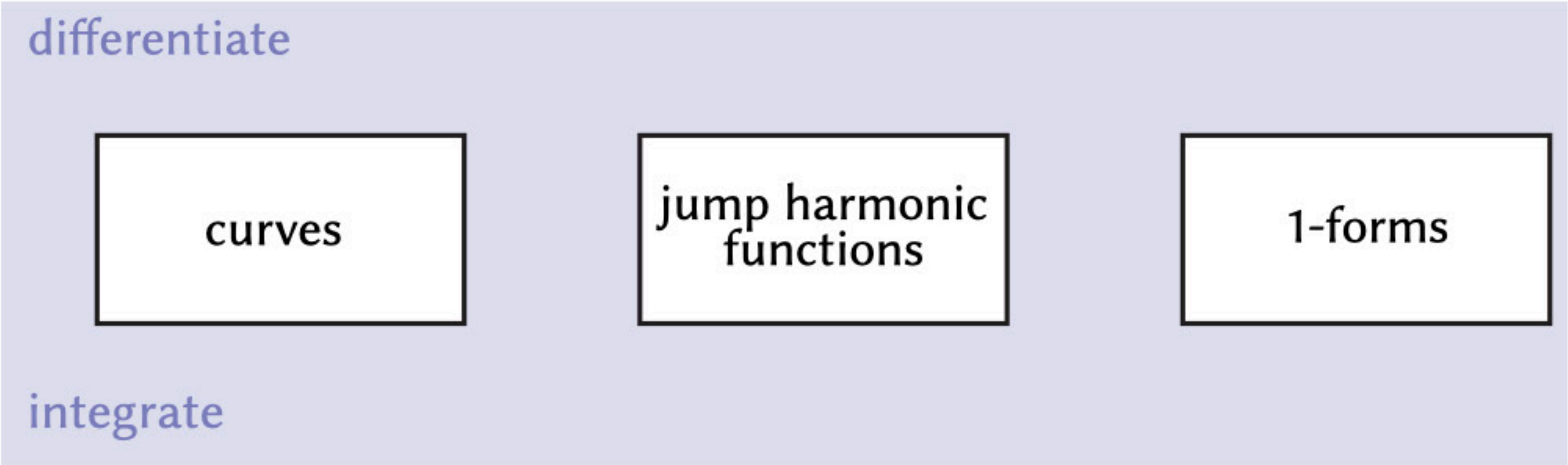
Solution: Solve for a new harmonic function w with jumps only across Γ.



Completing the round-trip



Completing the round-trip



Completing the round-trip

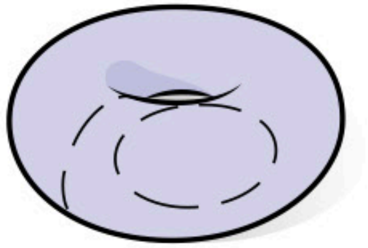
differentiate

curves

jump harmonic functions

1-forms

integrate

Input: 

Γ

Completing the round-trip

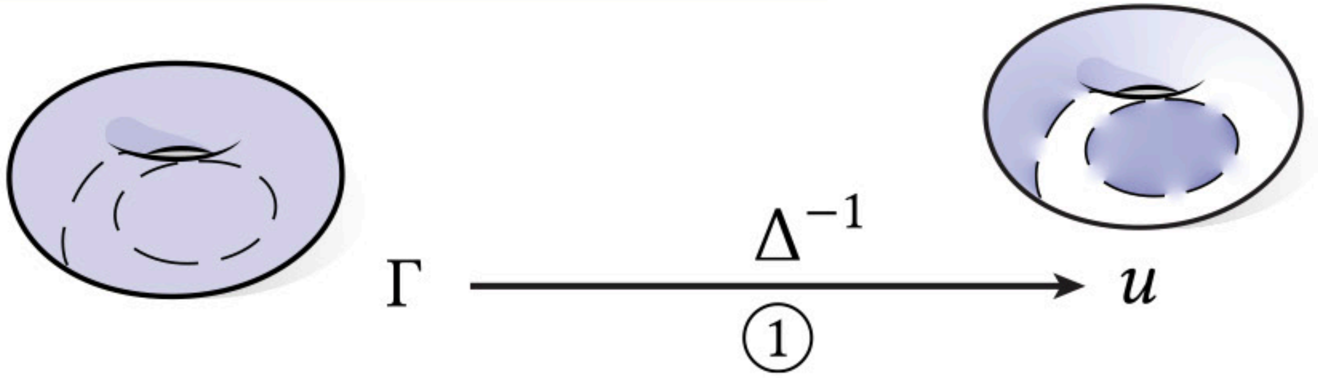
differentiate

curves → jump harmonic functions → 1-forms

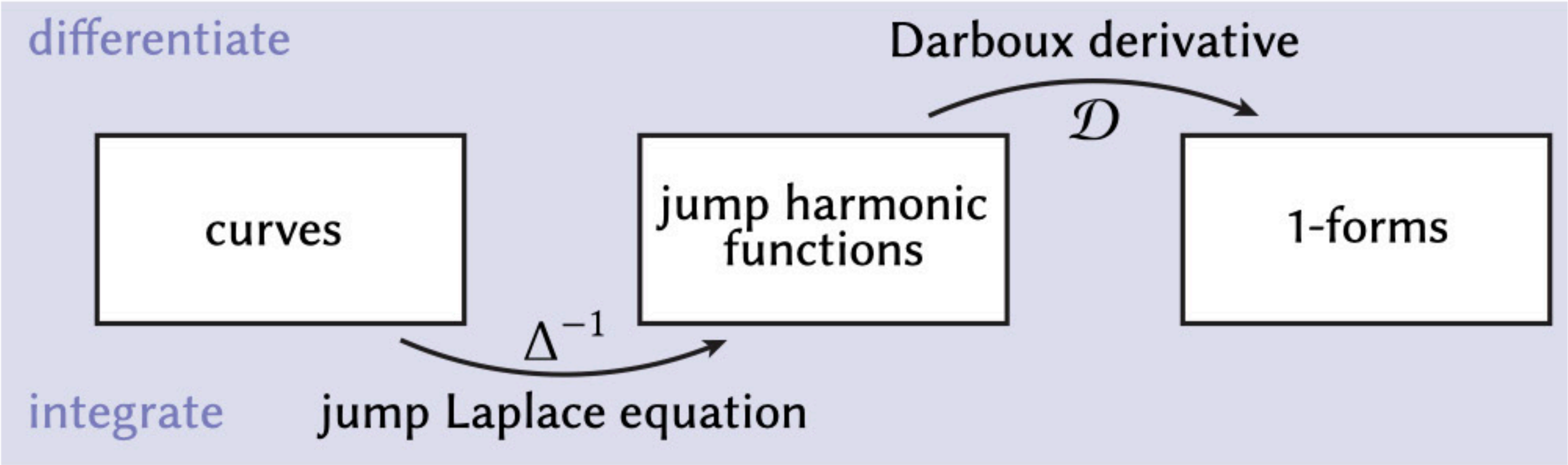
Δ^{-1}

integrate — jump Laplace equation

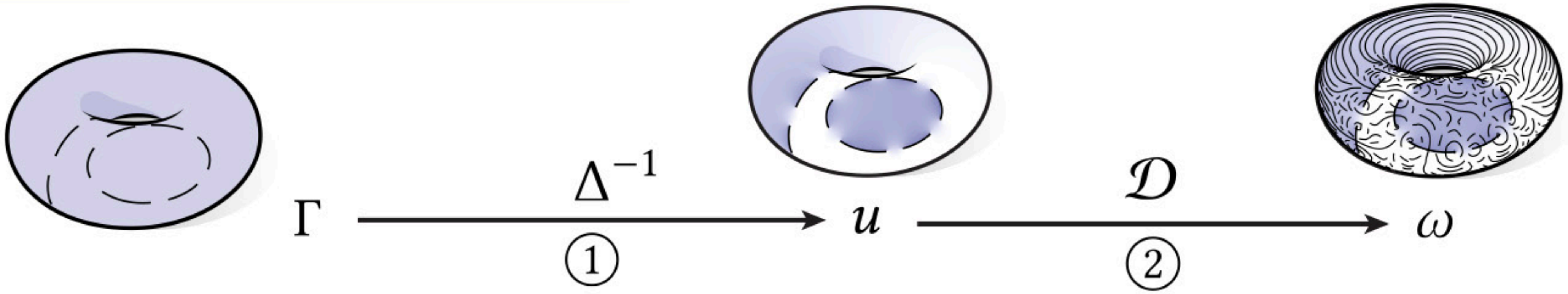
Input:



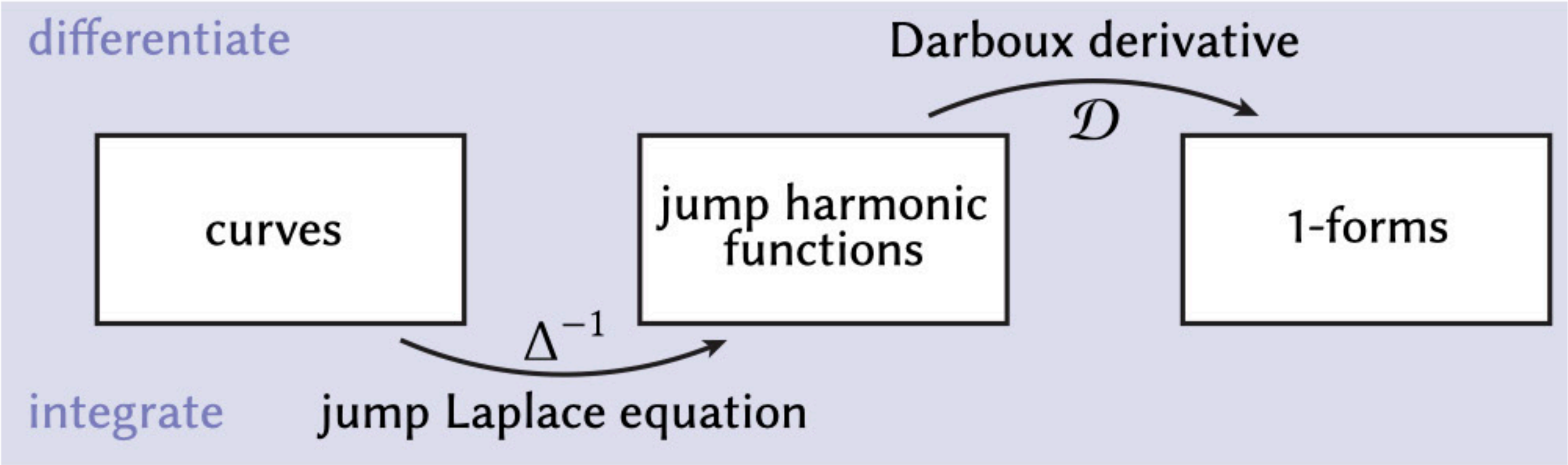
Completing the round-trip

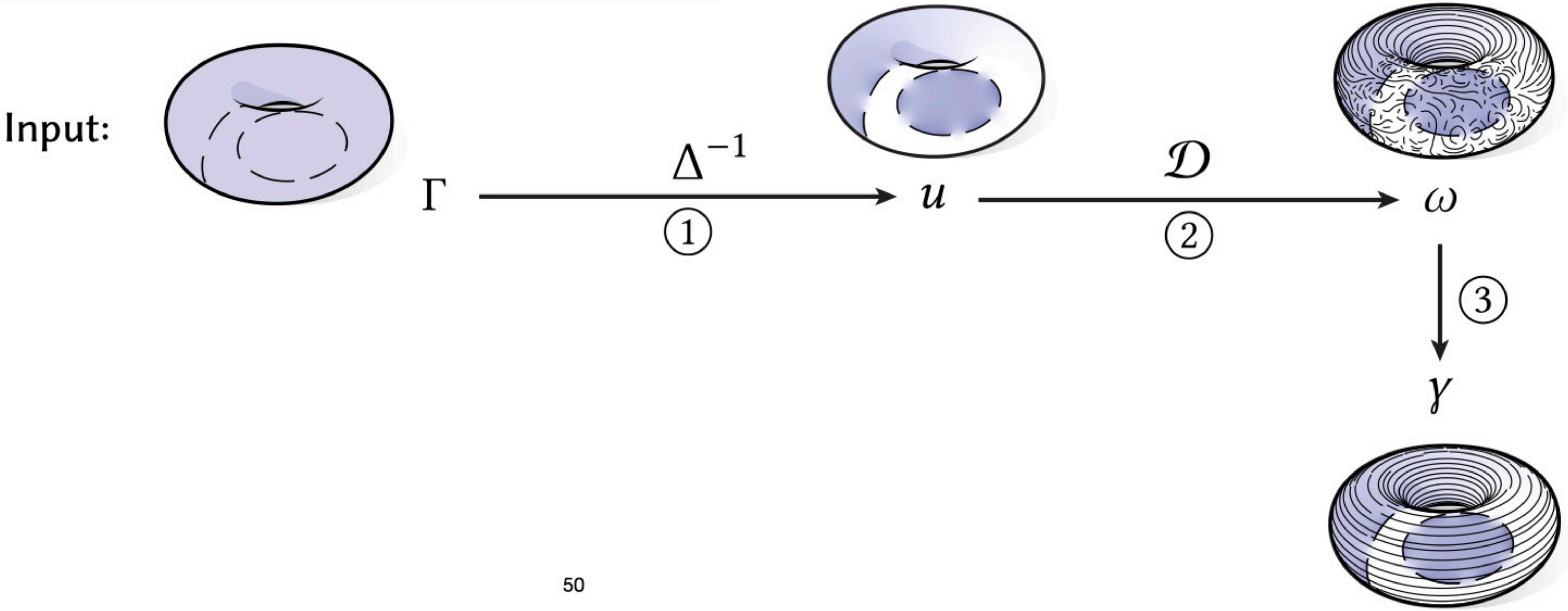


Input:

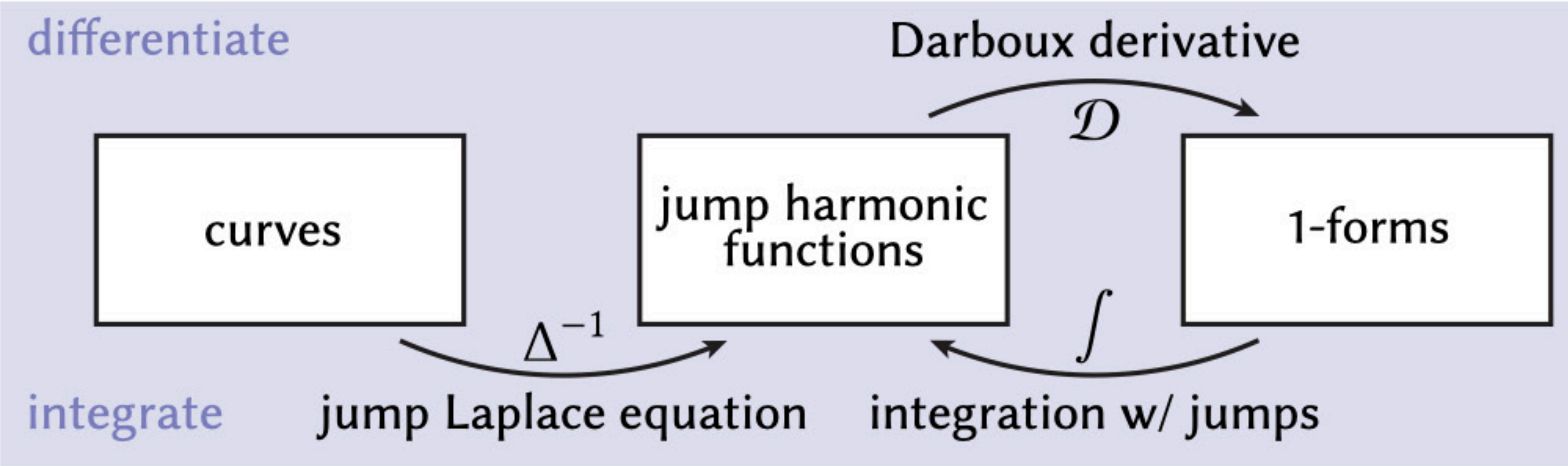


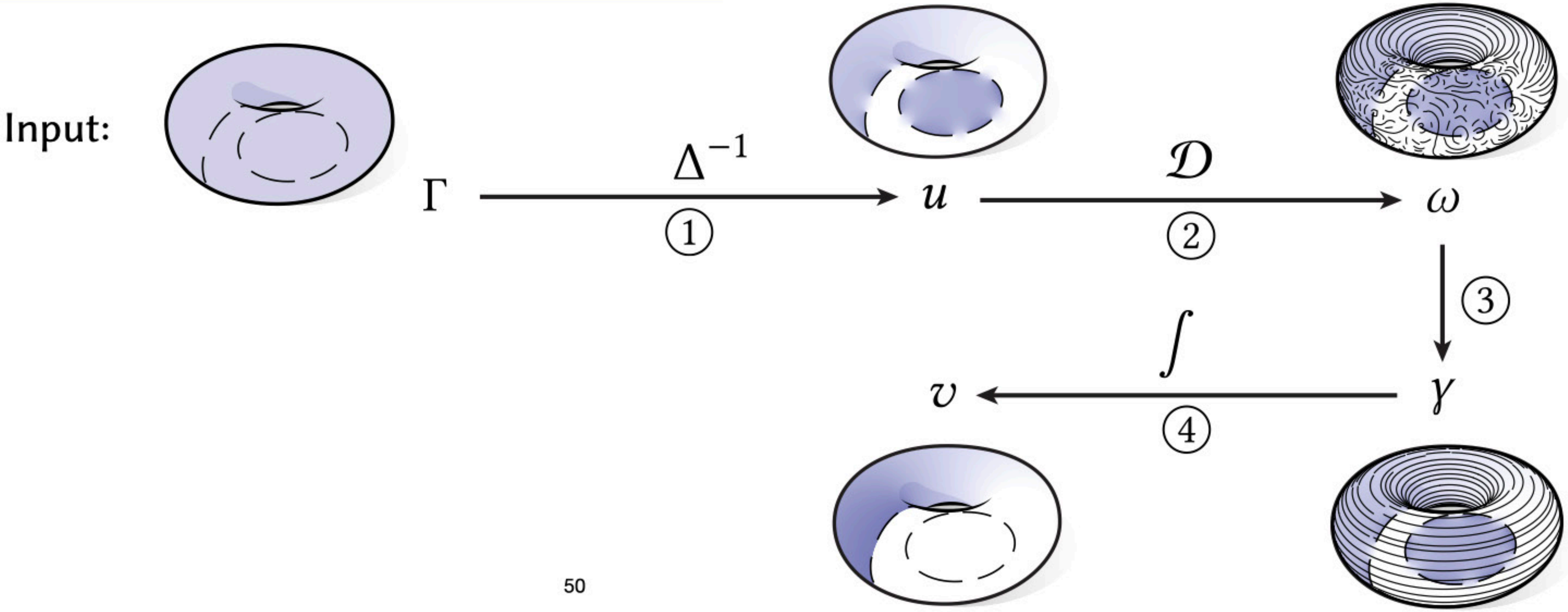
Completing the round-trip



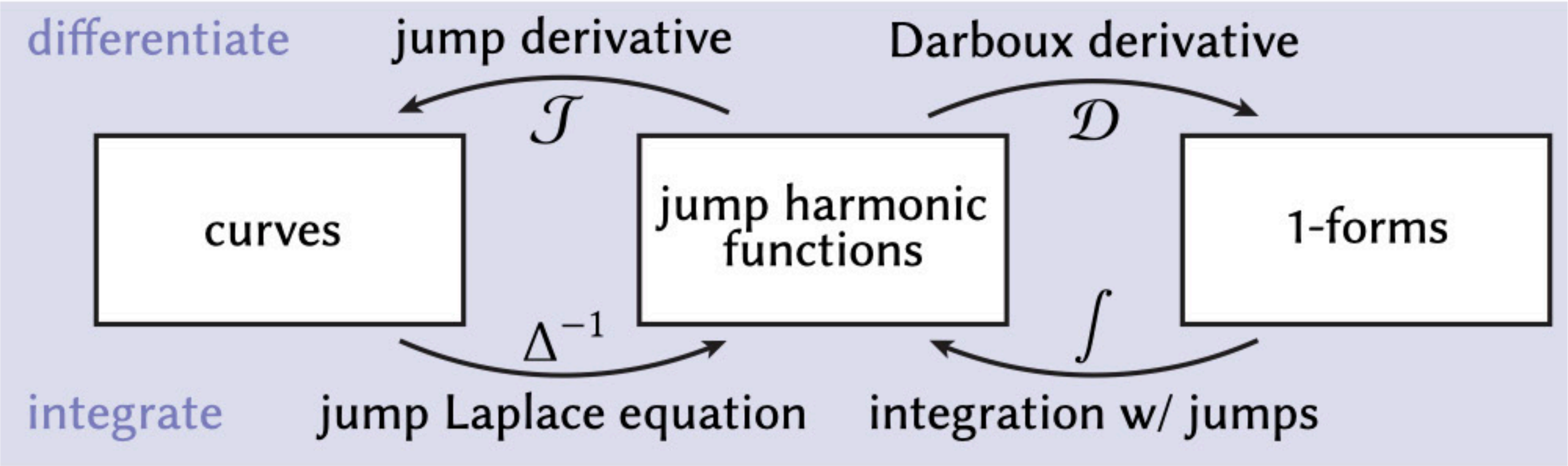


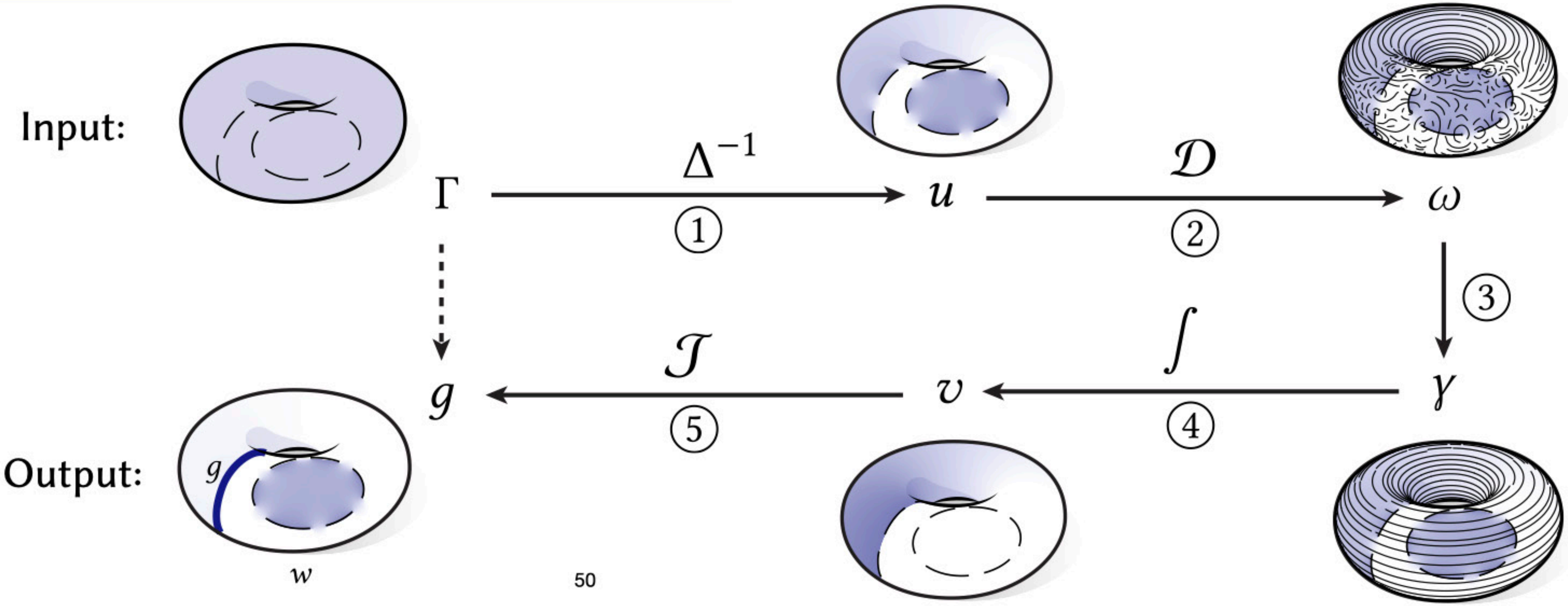
Completing the round-trip



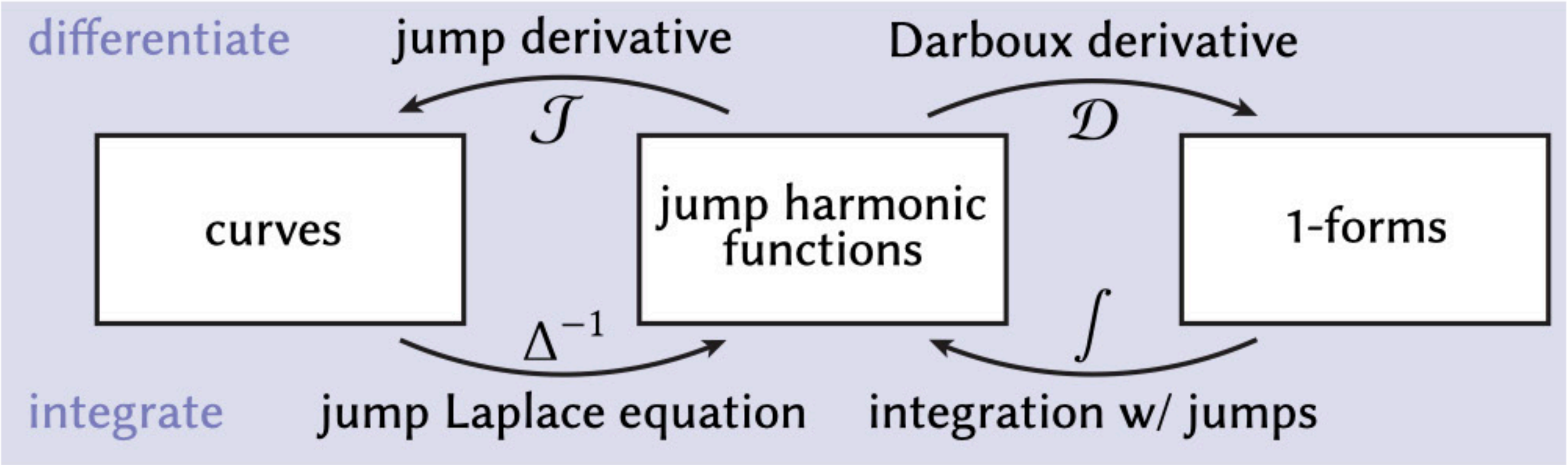


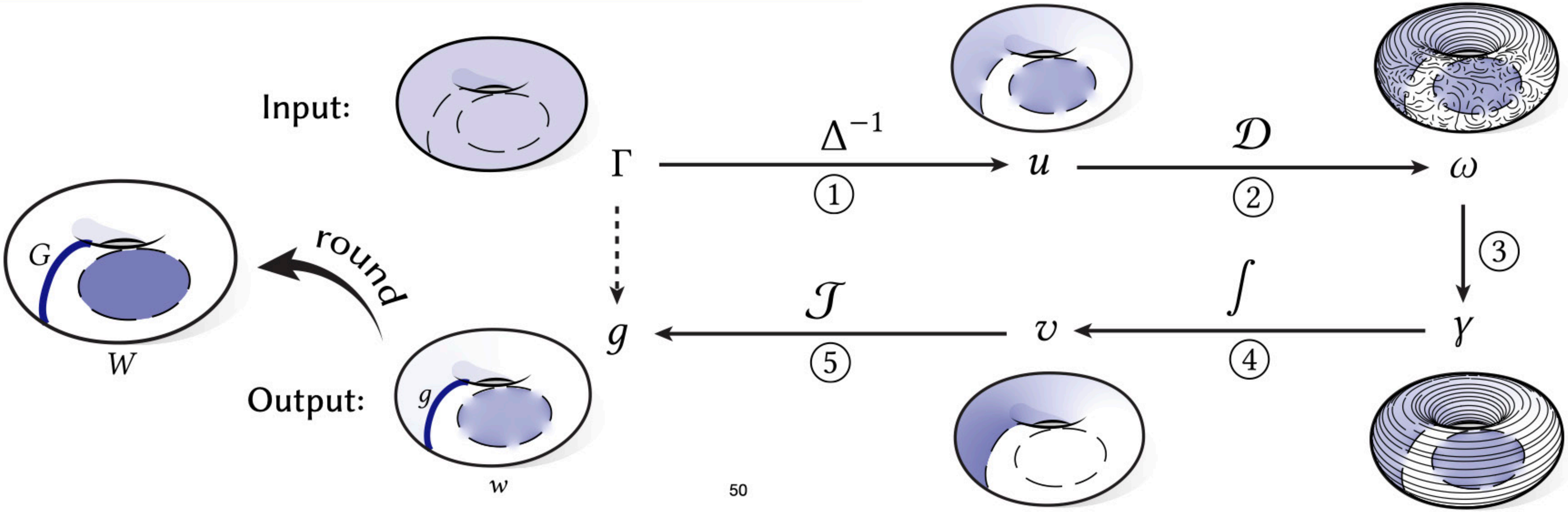
Completing the round-trip





Completing the round-trip





Algorithm summary

Algorithm summary

- Solve for a harmonic function u with jumps Γ.

Algorithm summary

- Solve for a harmonic function u with jumps Γ.
- Compute the harmonic part γ of $\mathcal{D}u$.

Algorithm summary

- Solve for a harmonic function u with jumps Γ.
- Compute the harmonic part γ of $\mathcal{D}u$.
- Solve a (linear) optimization to obtain the residual function v.

Algorithm summary

- Solve for a harmonic function u with jumps Γ.
- Compute the harmonic part γ of $\mathcal{D}u$.
- Solve a (linear) optimization to obtain the residual function v.
- Solve for the winding number function w with jumps $\Gamma - \mathcal{J}v$

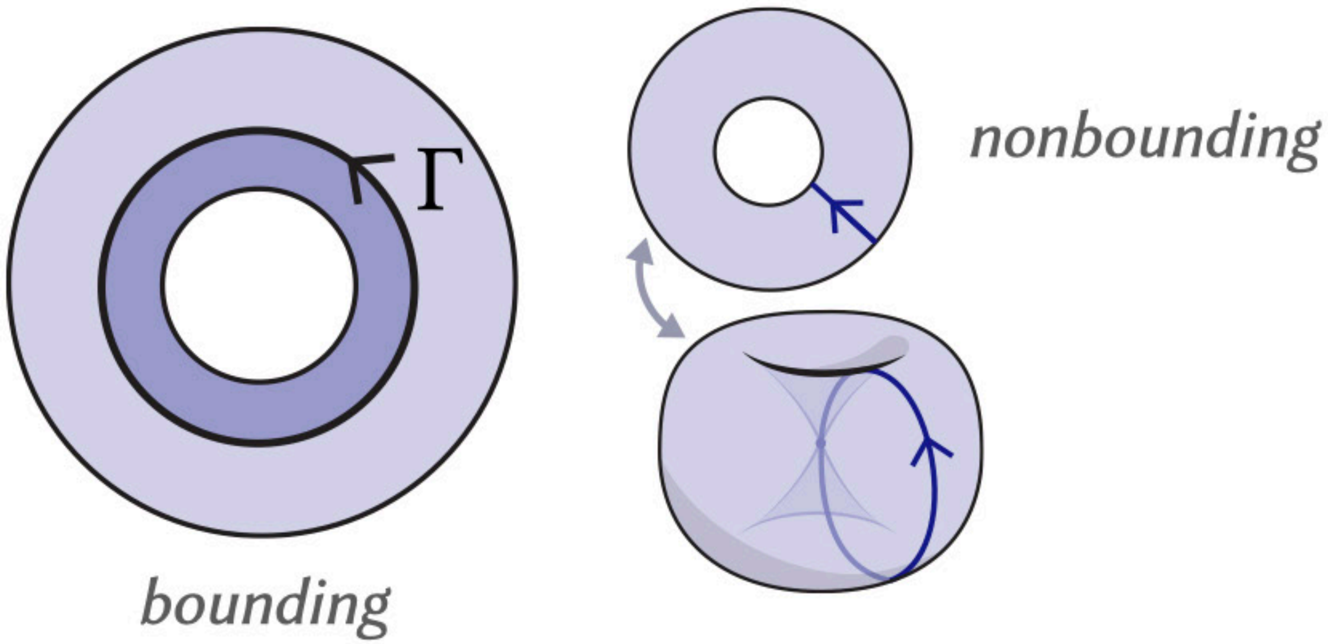
Algorithm summary

- Solve for a harmonic function u with jumps Γ.
- Compute the harmonic part γ of $\mathcal{D}u$.
- Solve a (linear) optimization to obtain the residual function v.
- Solve for the winding number function w with jumps $\Gamma - \mathcal{J}v$

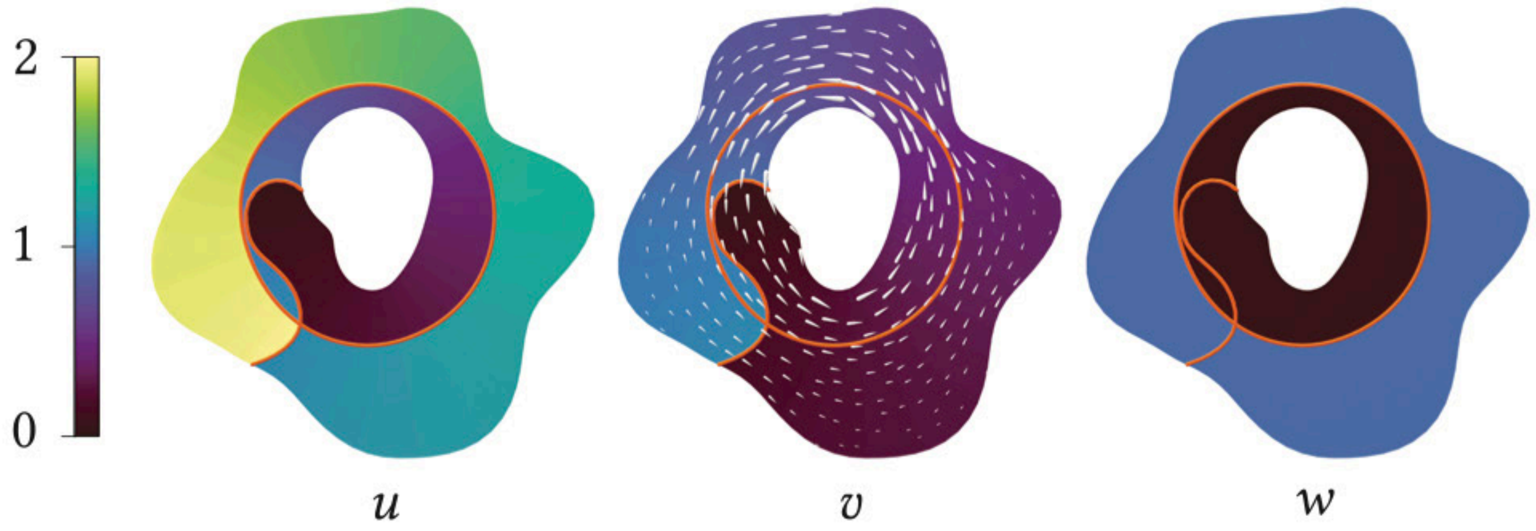
if surface M is multiply-connected

Surfaces with boundary

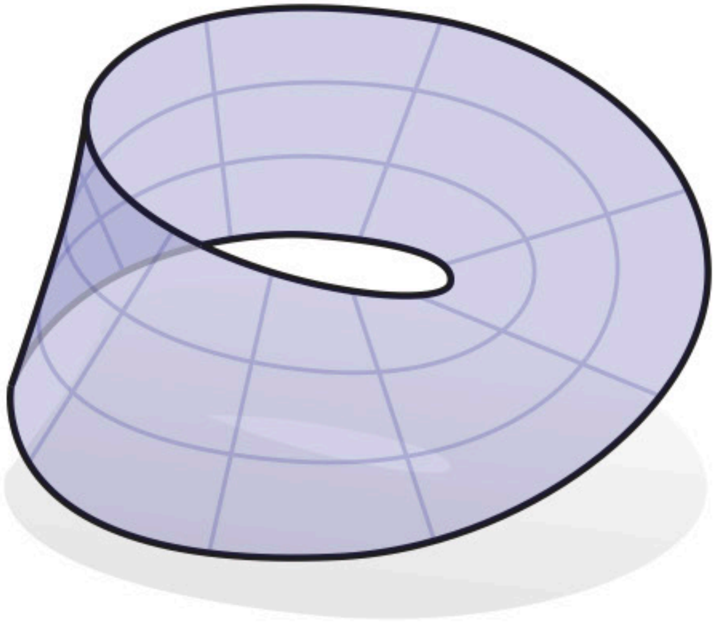
Bounding curves are those congruent to zero in the *relative homology* group $H_1(M, \partial M)$.



The rest of the theory follows.

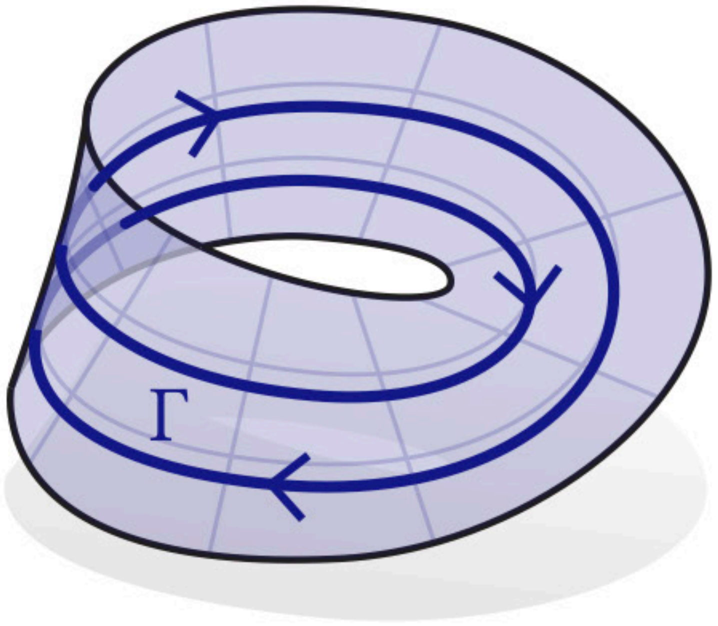


Non-orientable surfaces



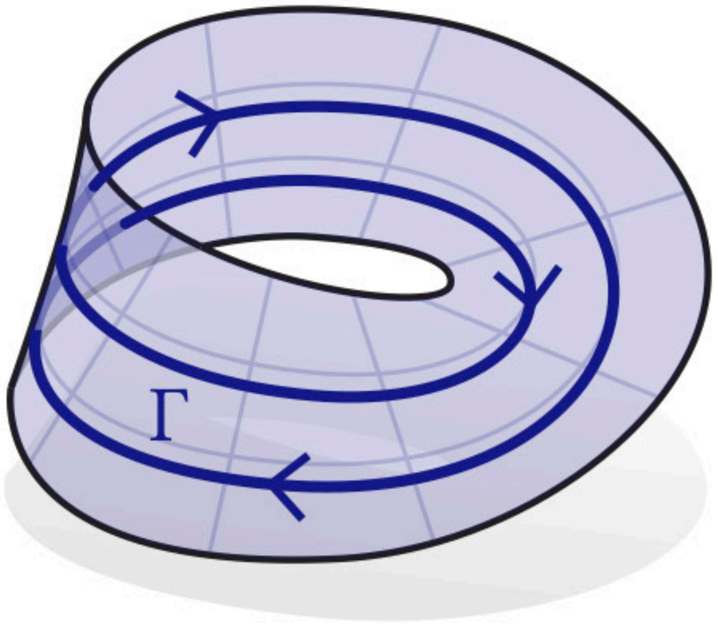
Non-orientable surfaces

orientable curve

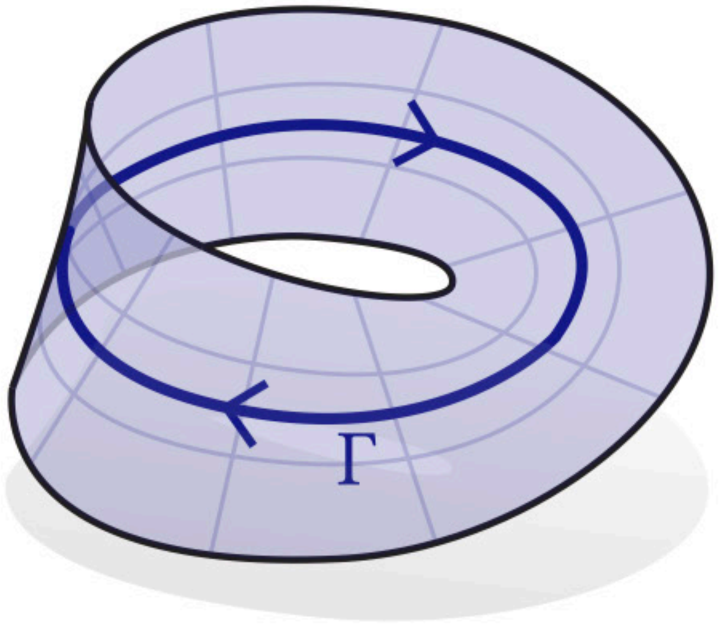


Non-orientable surfaces

orientable curve

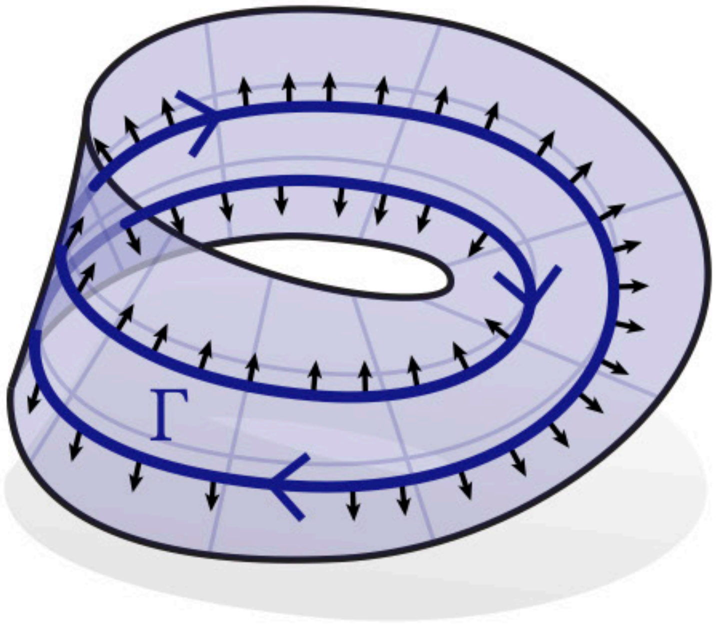


non-orientable curve

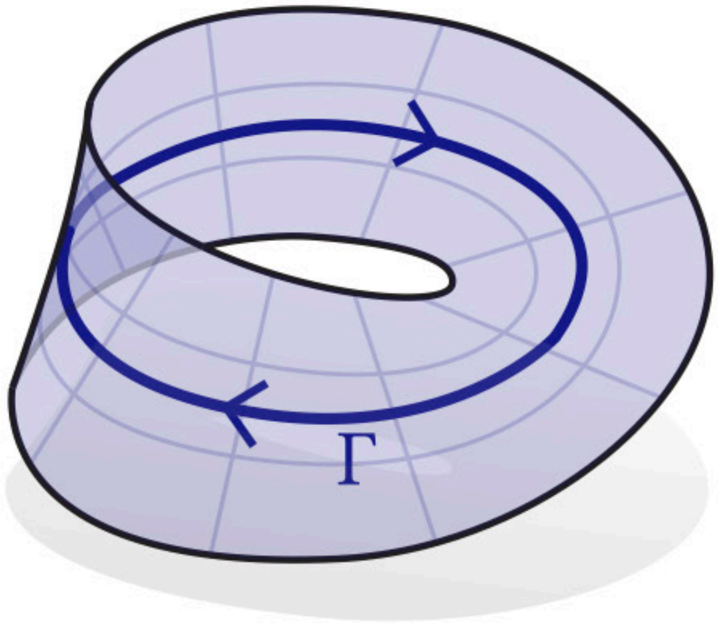


Non-orientable surfaces

orientable curve

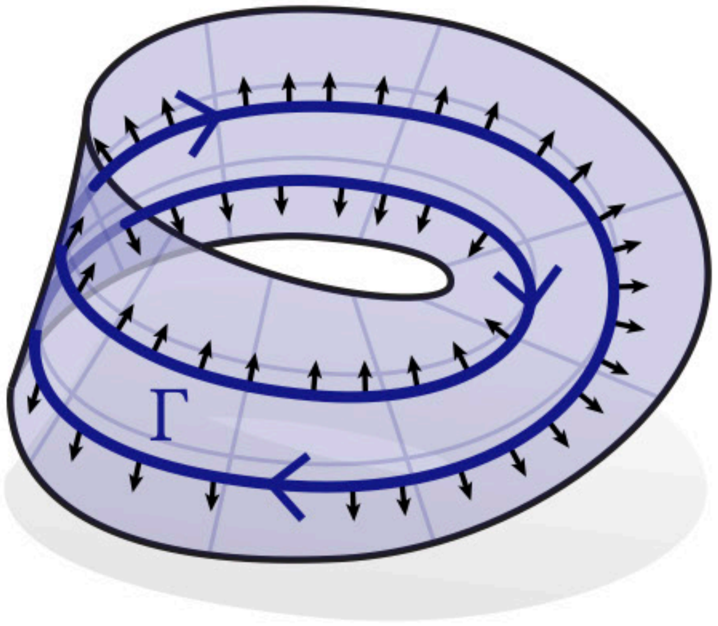


non-orientable curve

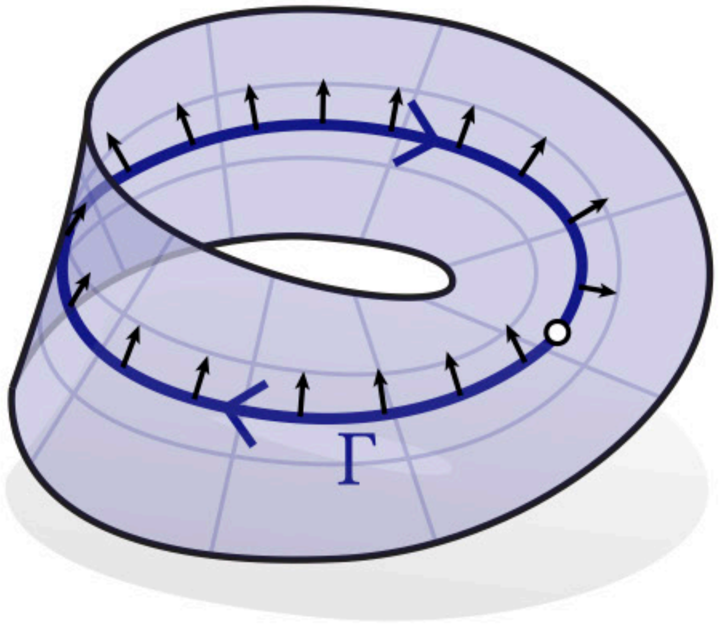


Non-orientable surfaces

orientable curve

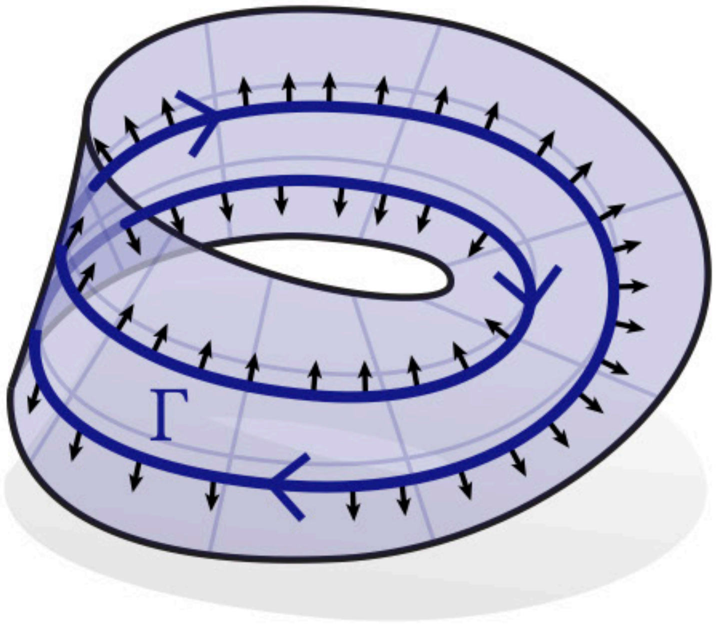


non-orientable curve

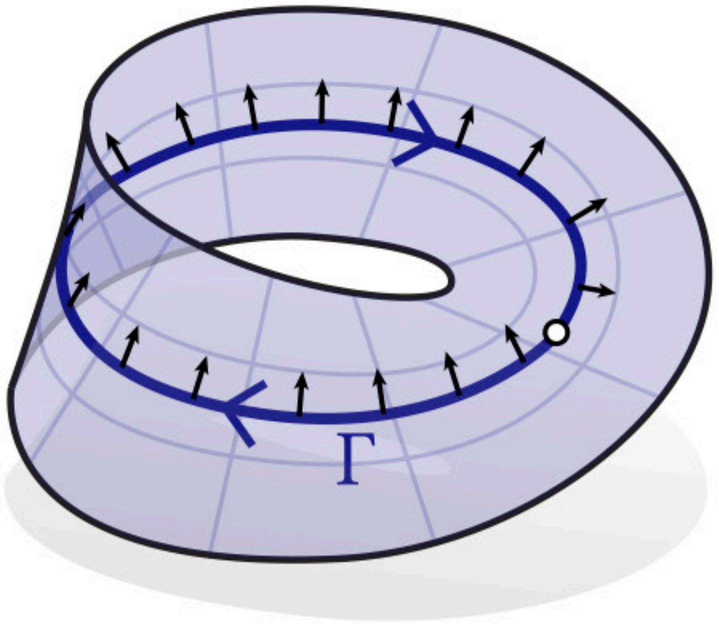


Non-orientable surfaces

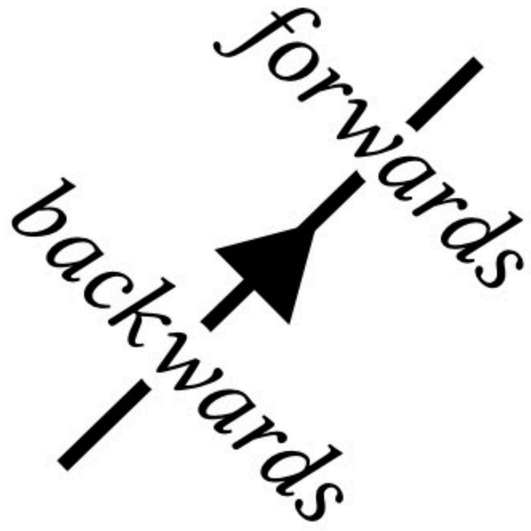
orientable curve



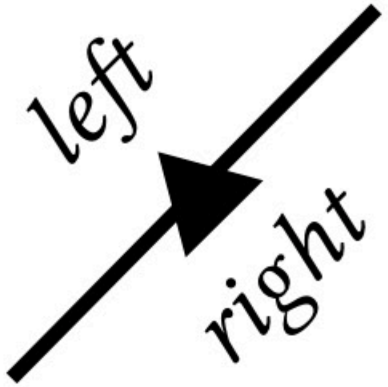
non-orientable curve



tangential

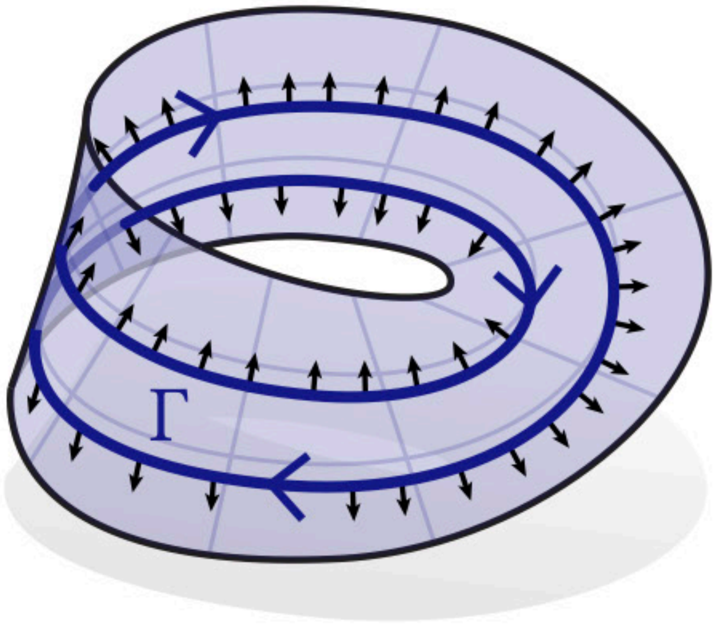


normal

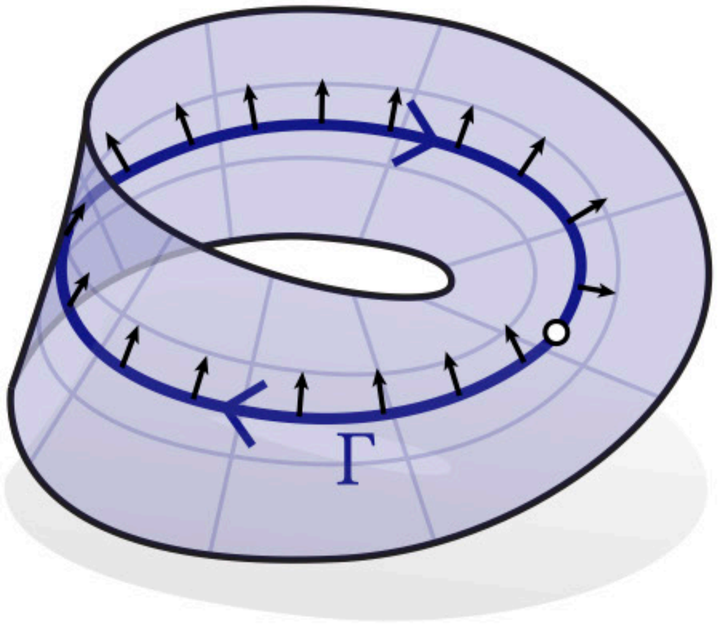


Non-orientable surfaces

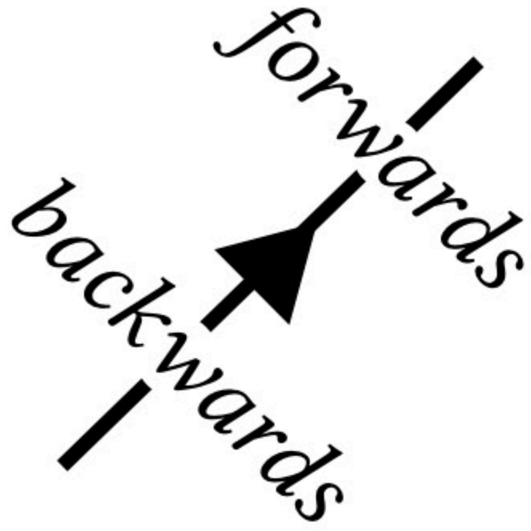
orientable curve



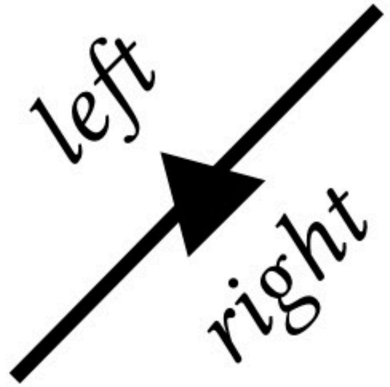
non-orientable curve



tangential



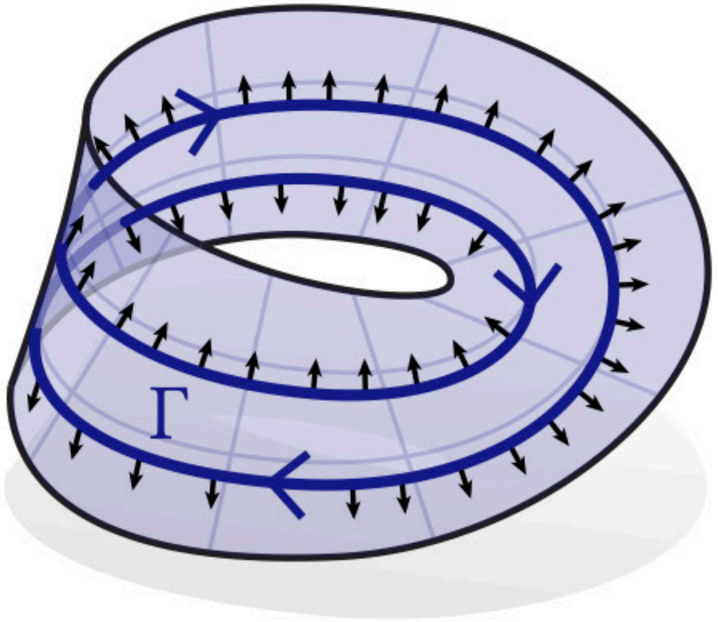
normal



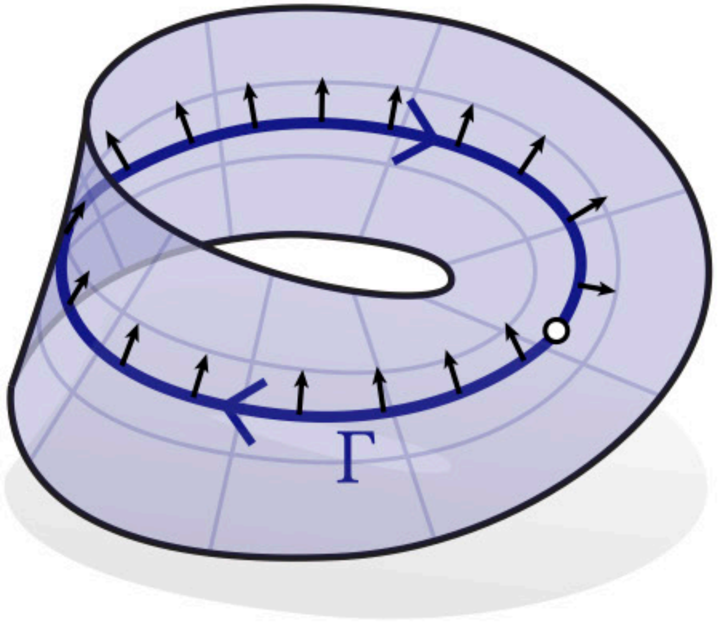
On orientable surfaces,
tangential orientation ≡ normal orientation

Non-orientable surfaces

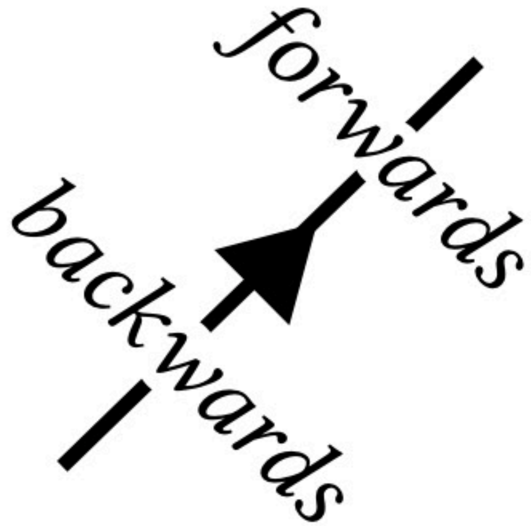
orientable curve



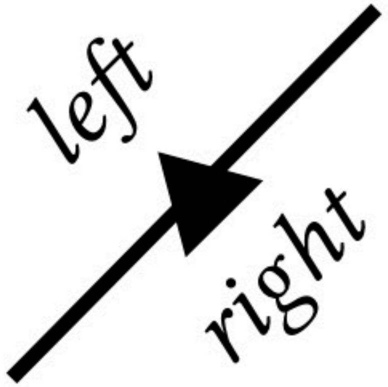
non-orientable curve



tangential



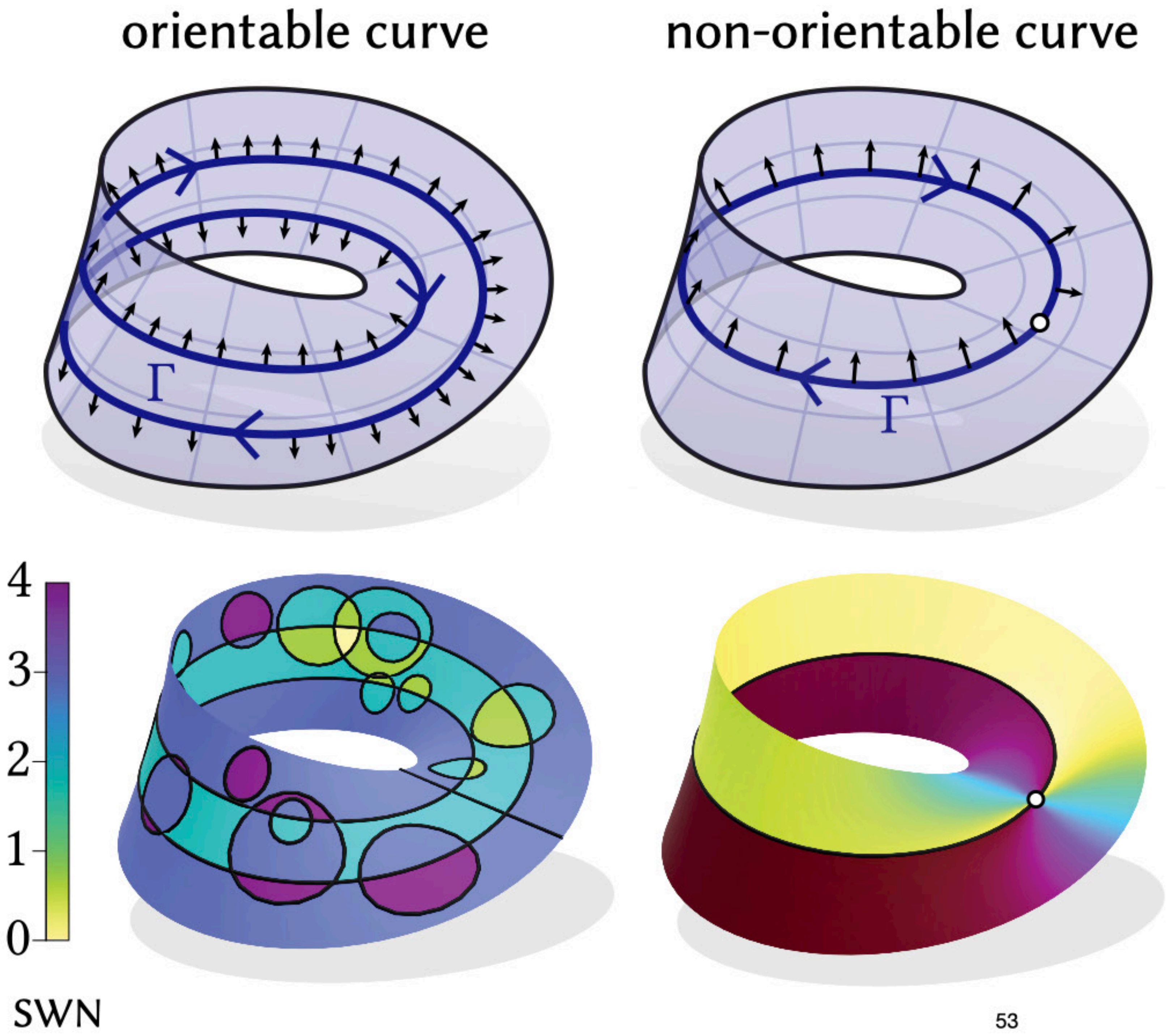
normal

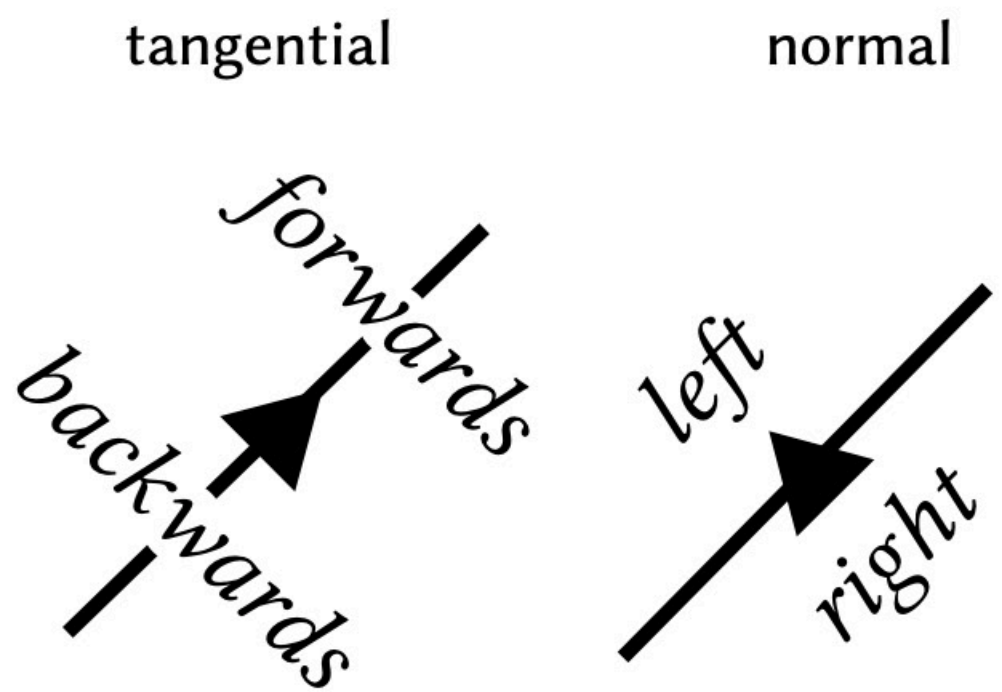


On orientable surfaces,
tangential orientation ≡ normal orientation

On **non**-orientable surfaces,
must specify normal orientation

Non-orientable surfaces



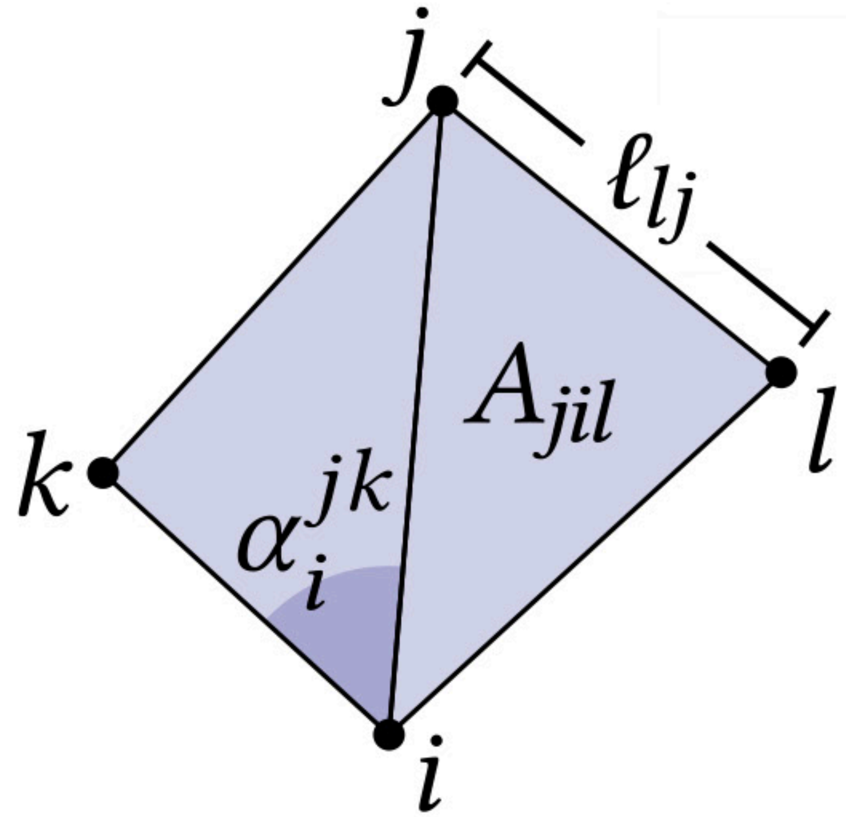


On orientable surfaces,
tangential orientation ≡ normal orientation

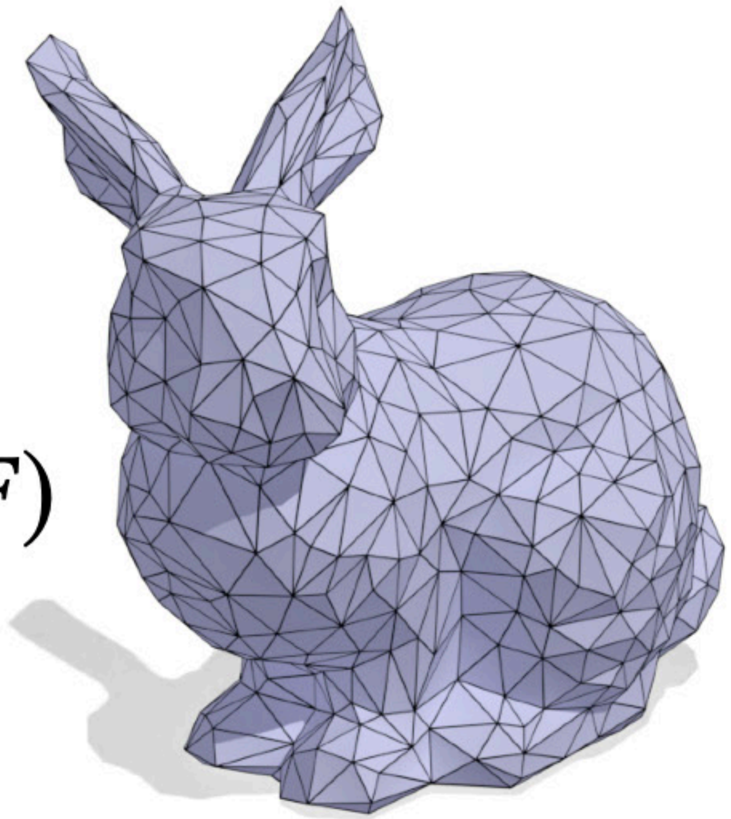
On **non**-orientable surfaces,
must specify normal orientation

DISCRETIZATION

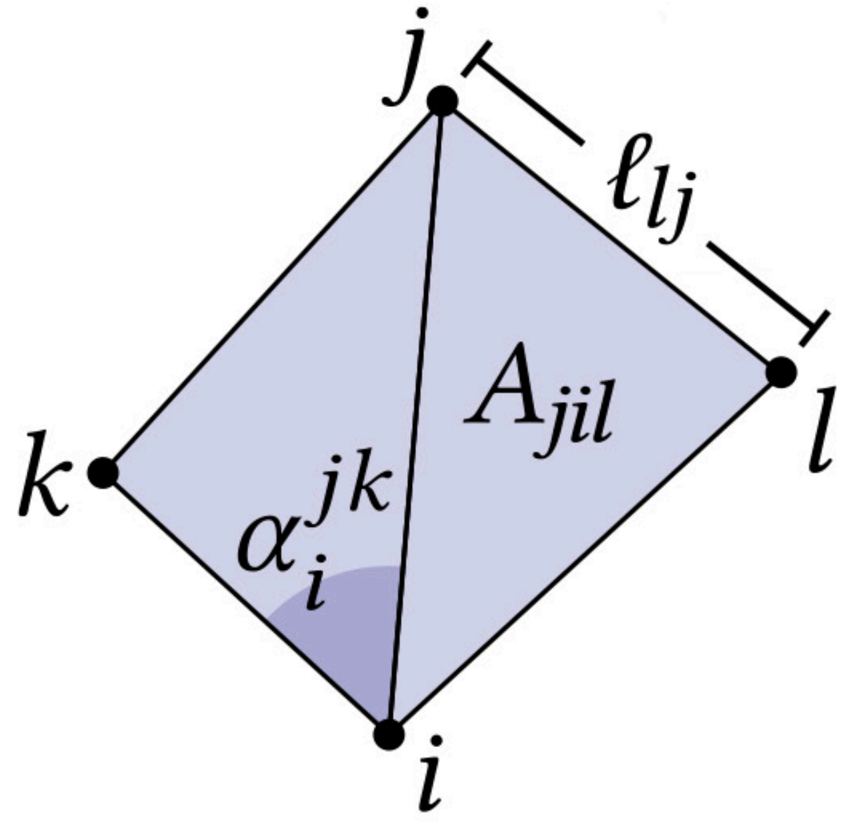
Curves & regions



$M = (V, E, F)$



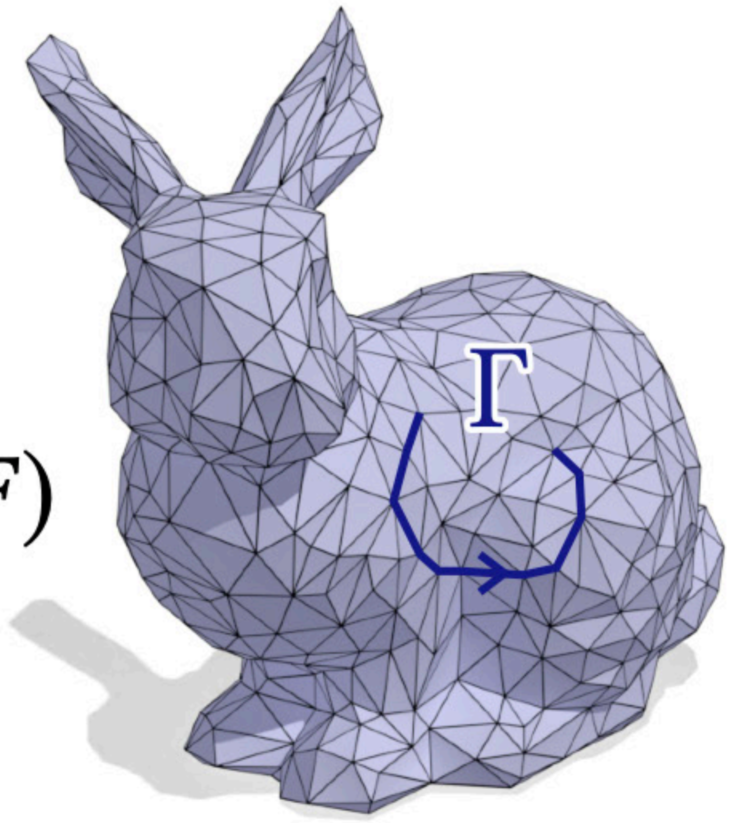
Curves & regions

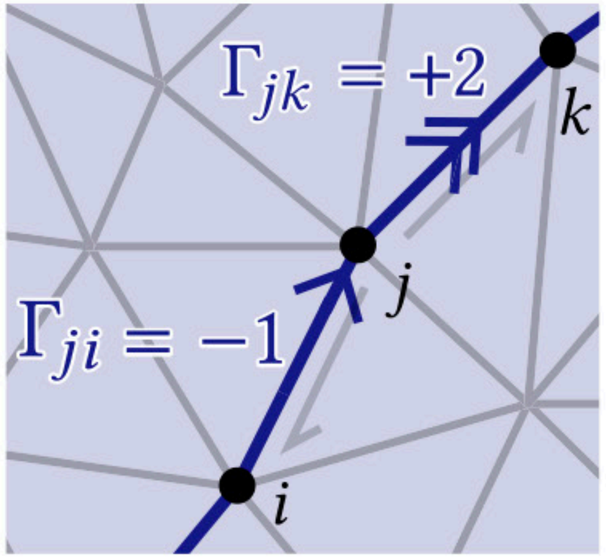


Γ is a **1-chain,** i.e. a signed integer per edge.

Regions are **2-chains,** signed integers per face.

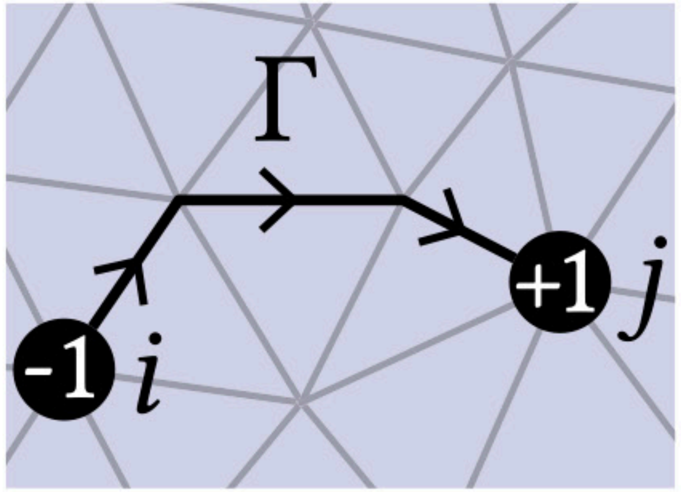
$$M = (V, E, F)$$





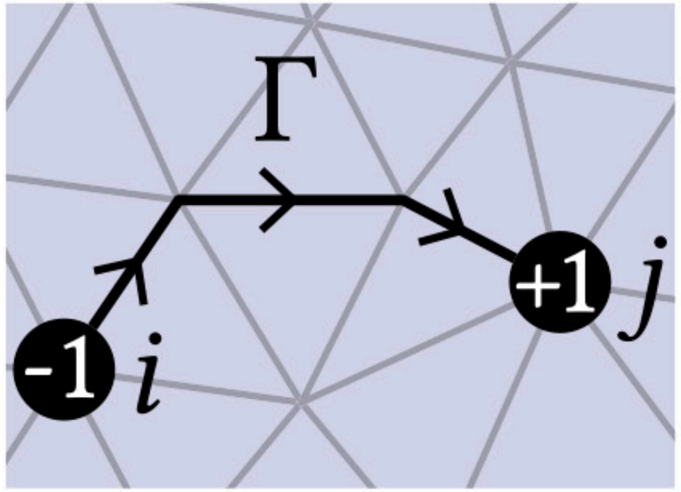
Endpoints

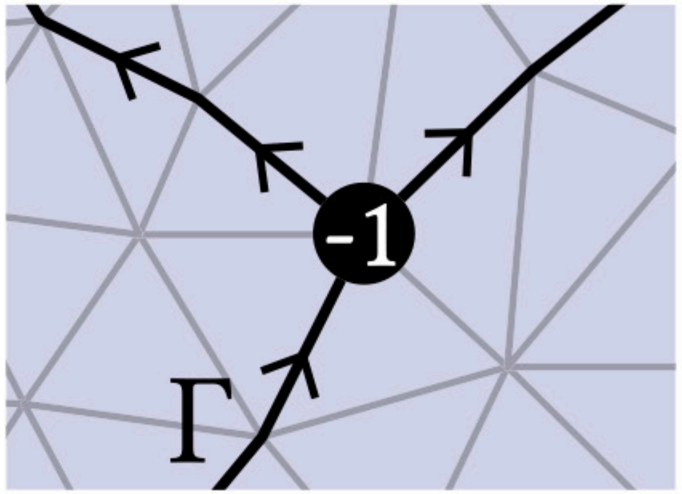
The boundary of Γ is a *0-chain,* $(\partial\Gamma)_i := -\Sigma_{ij}\Gamma_{ij}$



Endpoints

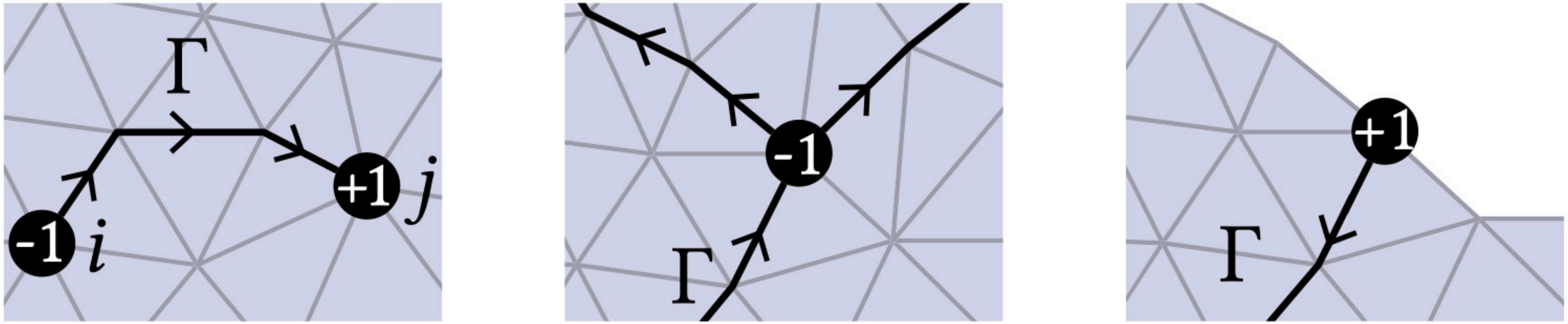
The boundary of Γ is a *0-chain,* $(\partial\Gamma)_i := -\Sigma_{ij}\Gamma_{ij}$





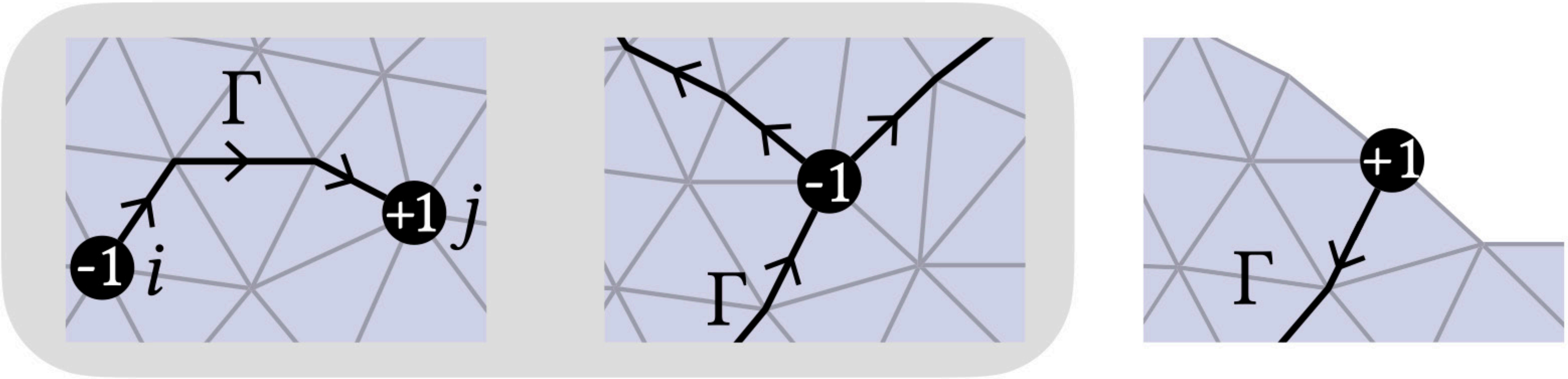
Endpoints

The boundary of Γ is a *0-chain,* $(\partial\Gamma)_i := -\Sigma_{ij}\Gamma_{ij}$



Endpoints

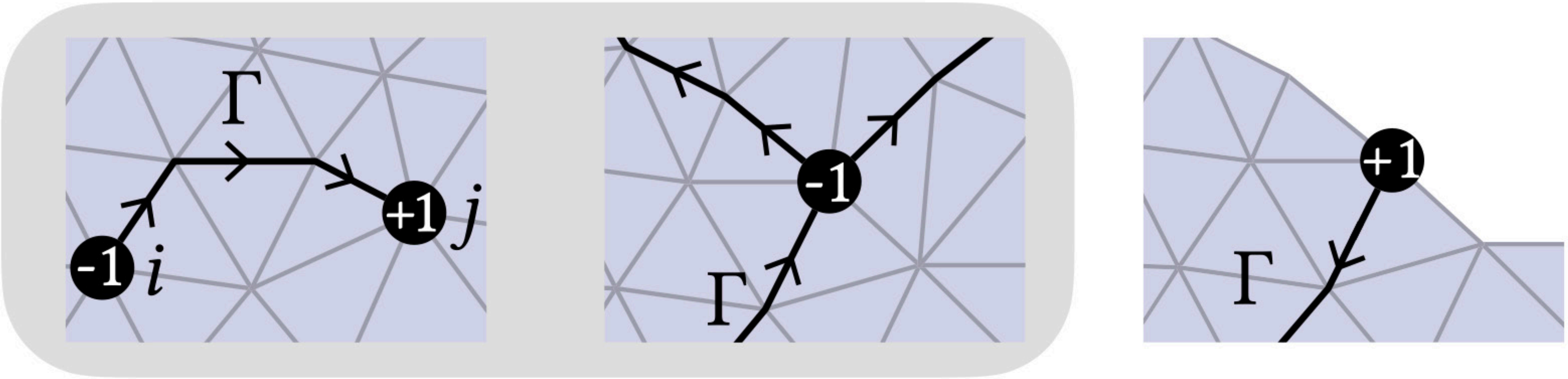
The boundary of Γ is a *0-chain,* $(\partial\Gamma)_i := -\Sigma_{ij}\Gamma_{ij}$



interior endpoints

Endpoints

The boundary of Γ is a *0-chain,* $(\partial\Gamma)_i := -\Sigma_{ij}\Gamma_{ij}$

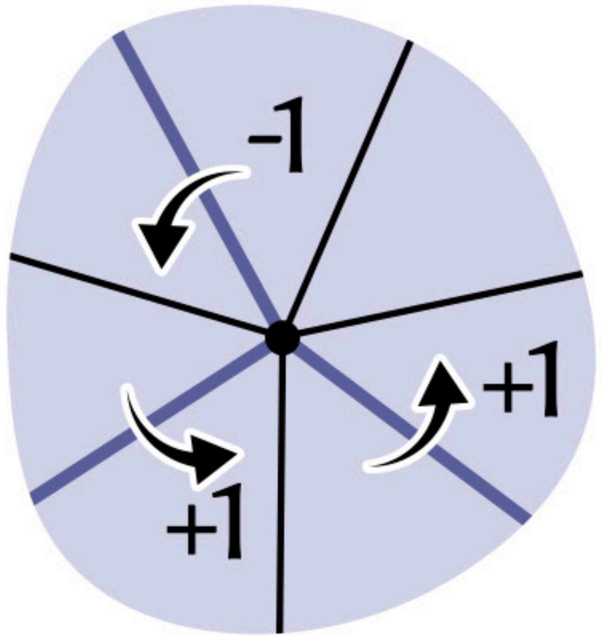


interior endpoints

V^* := set of mesh vertices that are not interior endpoints

E^* := set of edges with both points in V^*

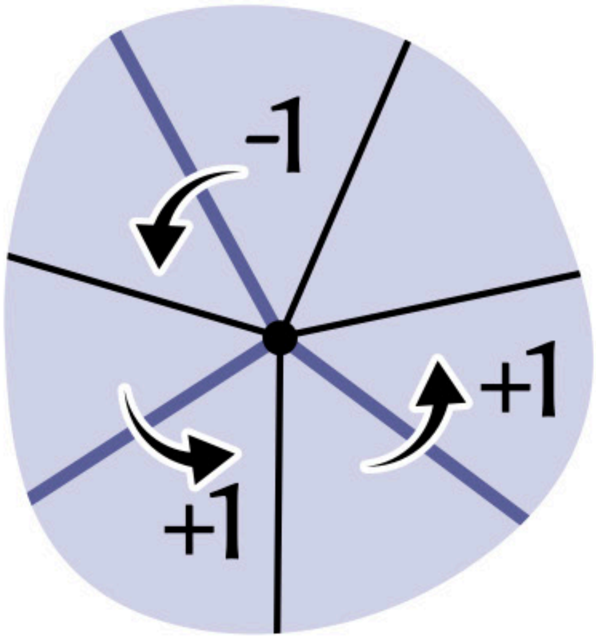
Endpoints are singular



Endpoints represent *singular points:*

There are no corner values compatible with jumps.

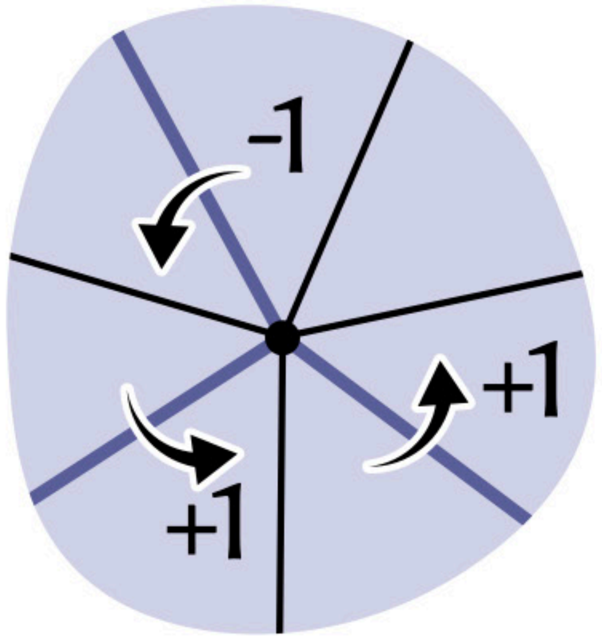
Endpoints are singular



Endpoints represent *singular points:*

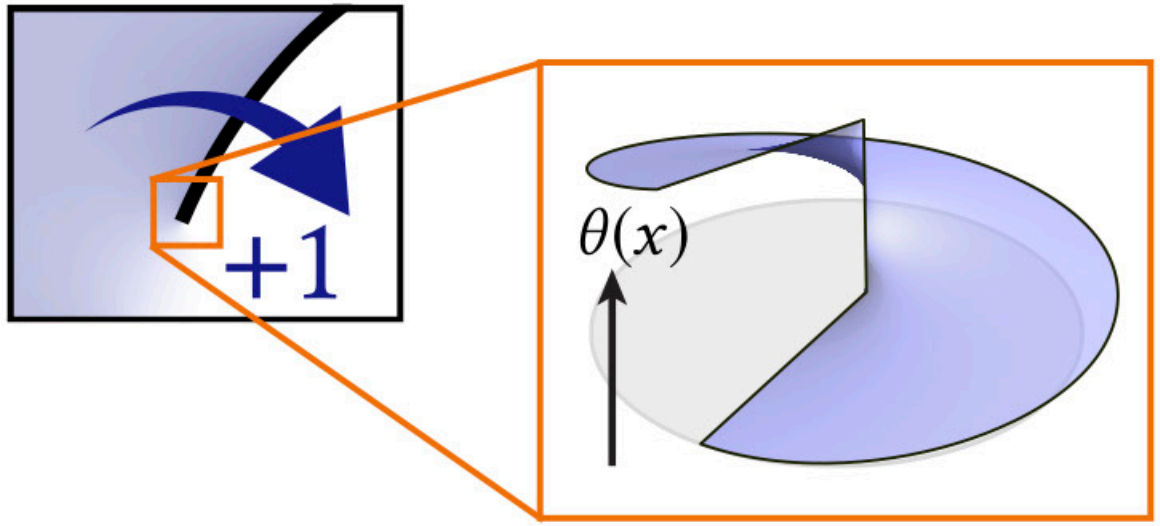
There are no corner values compatible with jumps.

Endpoints are singular

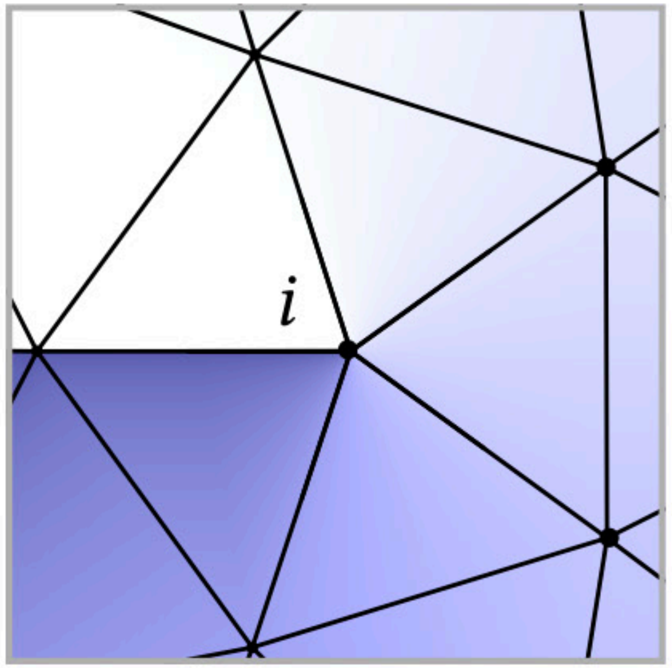


Endpoints represent *singular points:*

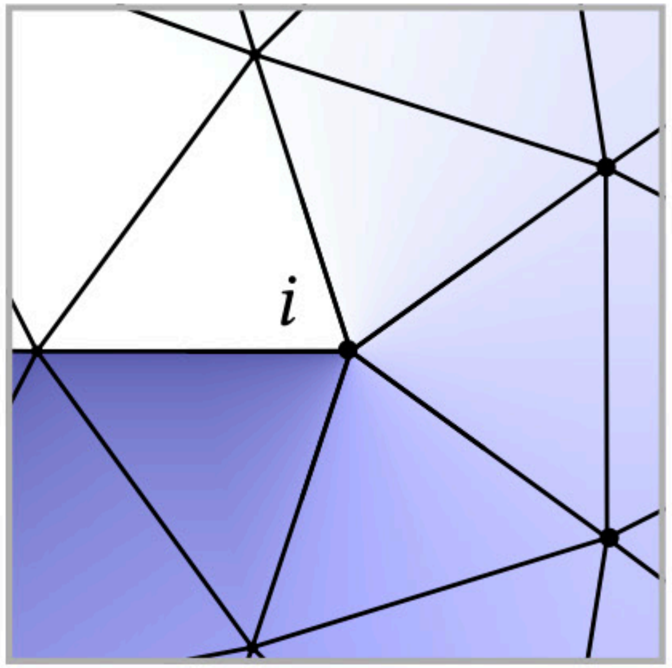
There are no corner values compatible with jumps.

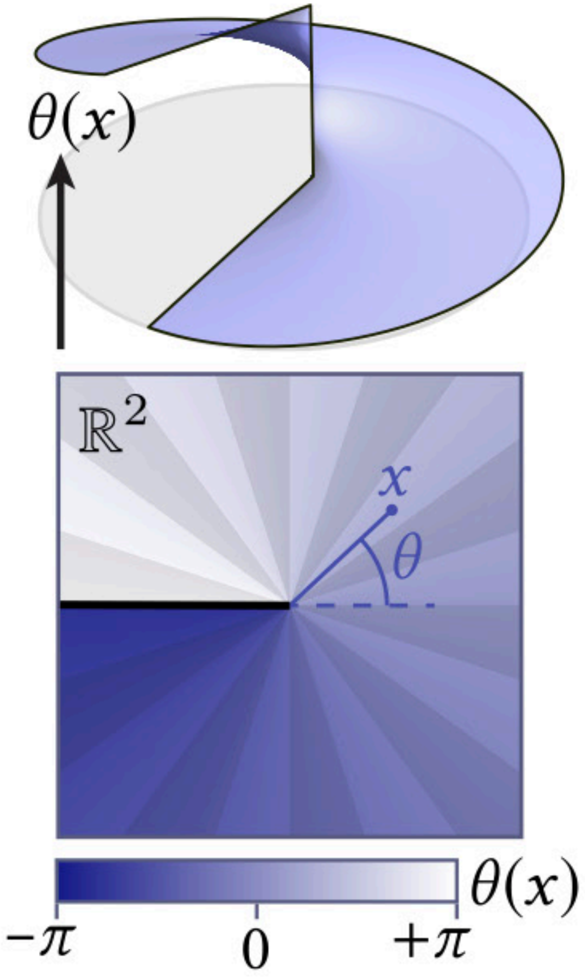


Singular point interpolation

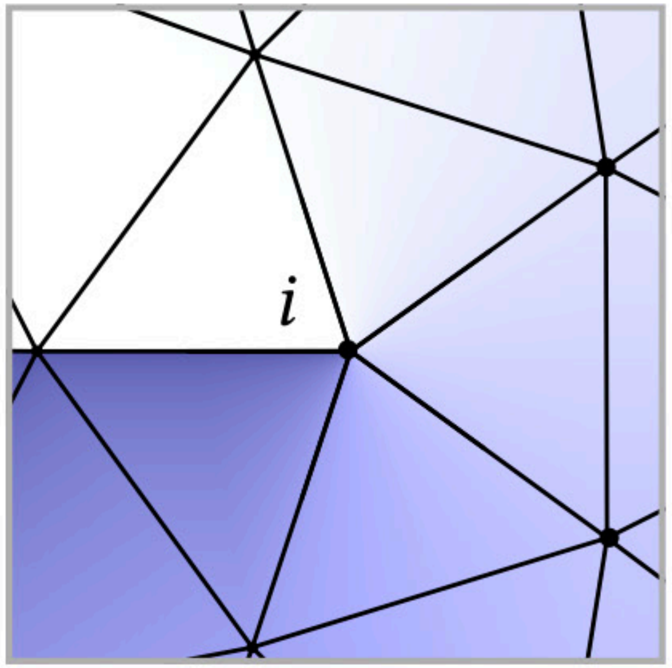


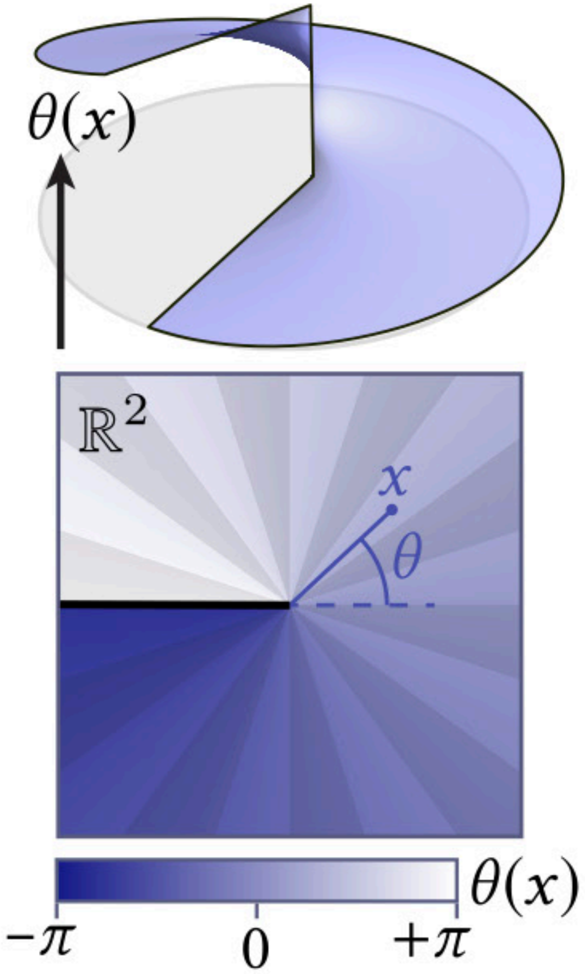
Singular point interpolation

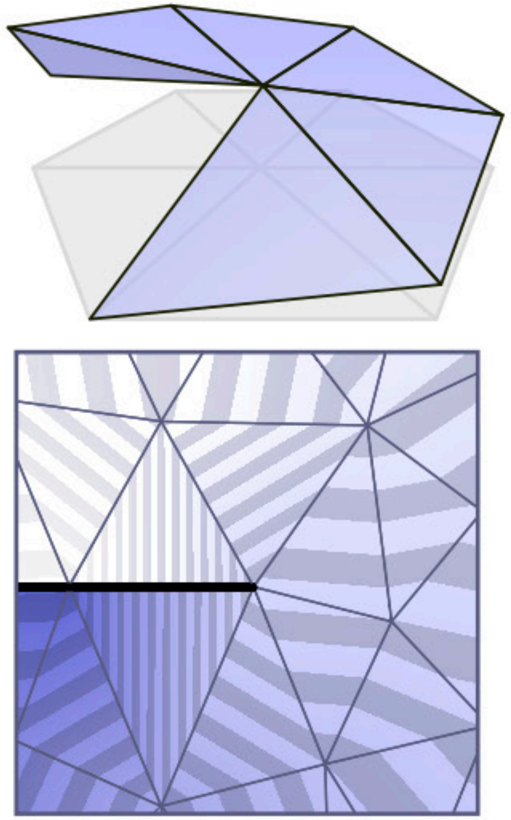




Singular point interpolation



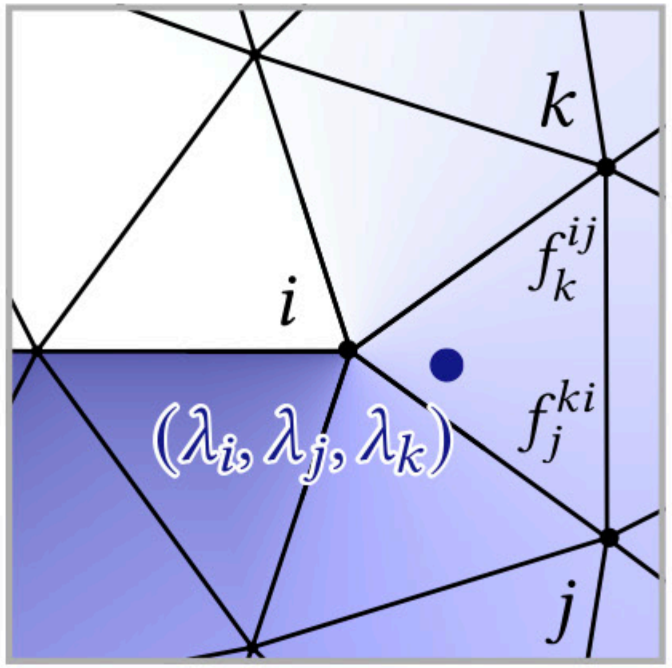


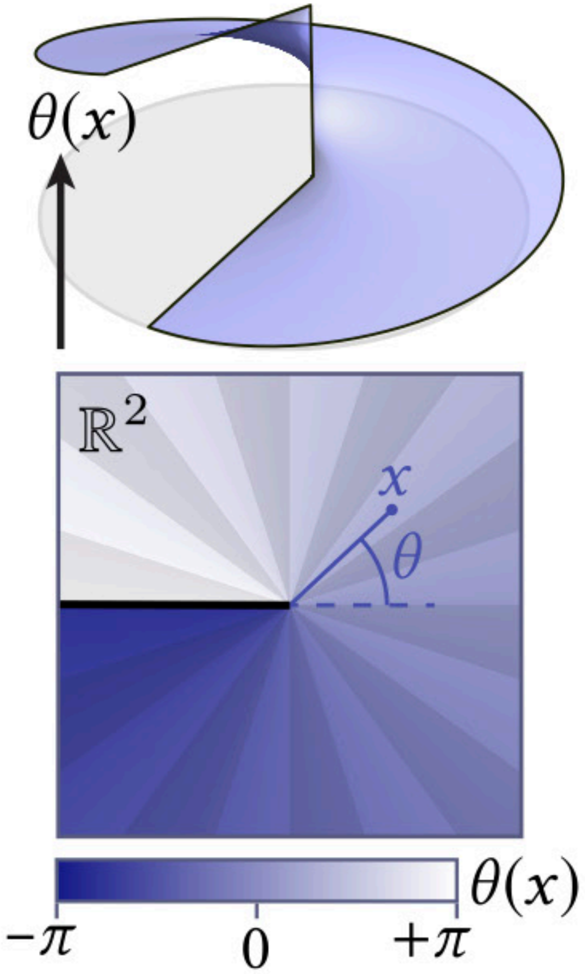


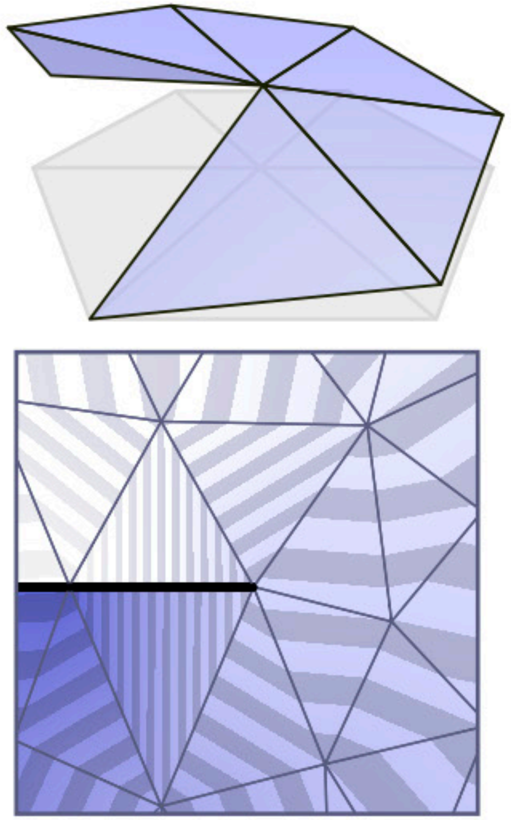
piecewise-linear interpolation of θ

Singular point interpolation

$$f(\lambda_i, \lambda_j, \lambda_k) := \frac{\lambda_j f_j^{ki} + \lambda_k f_k^{ij}}{\lambda_j + \lambda_k}$$



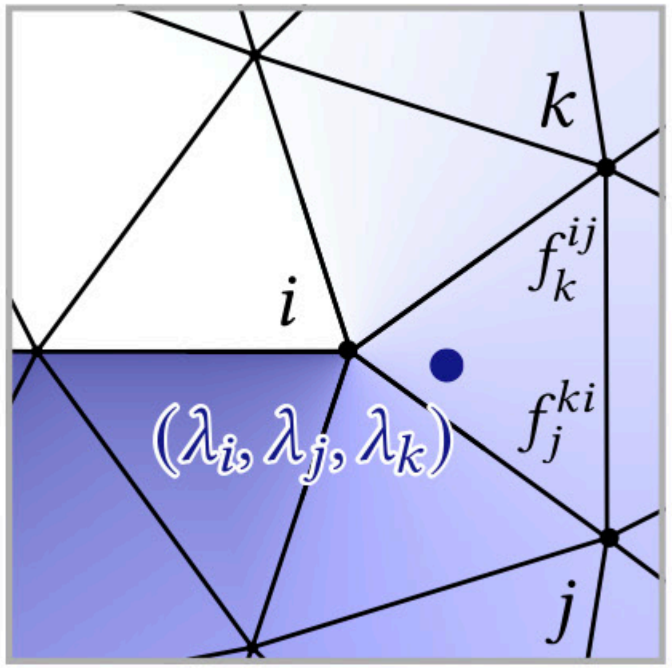


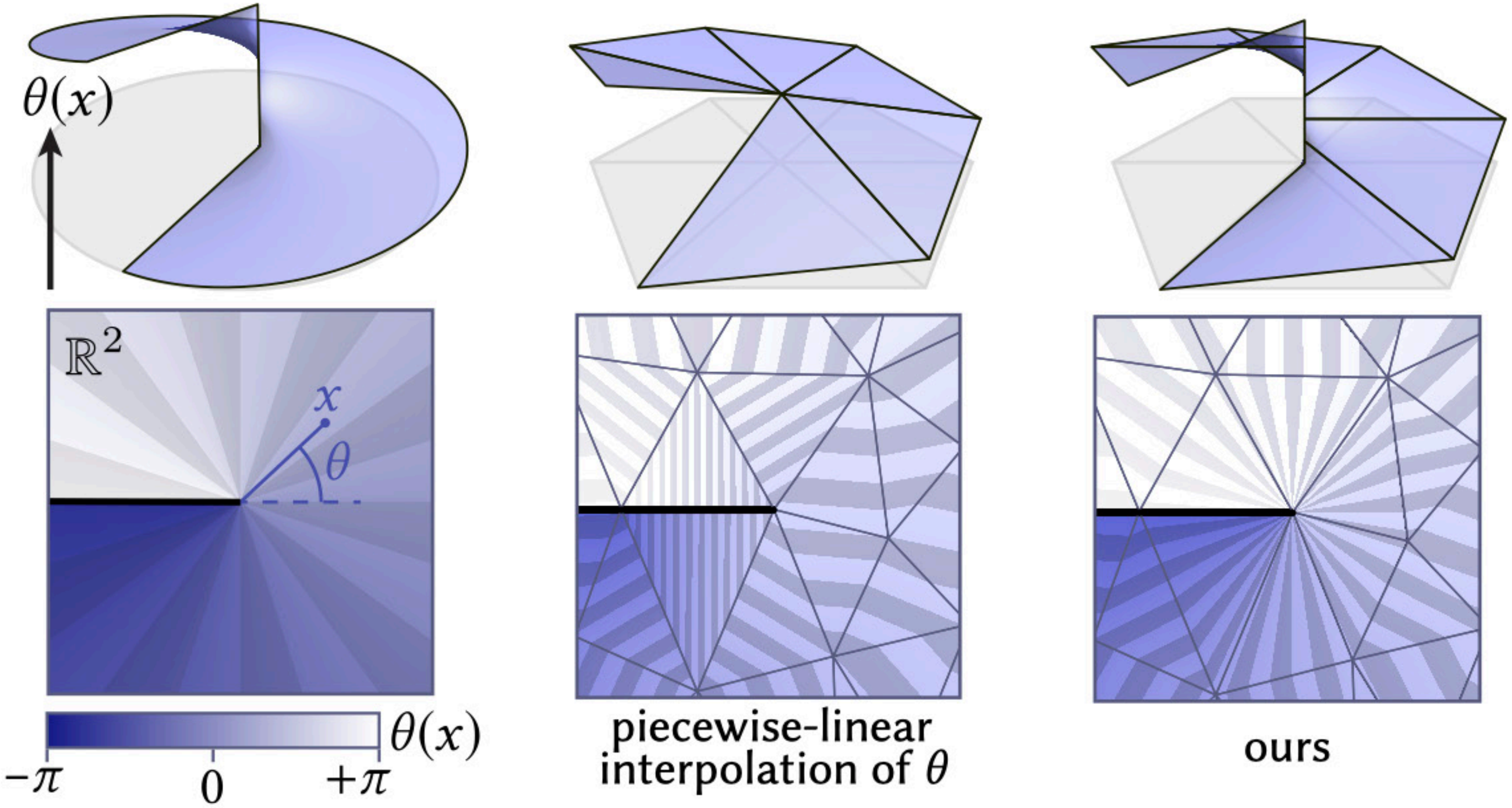


piecewise-linear interpolation of θ

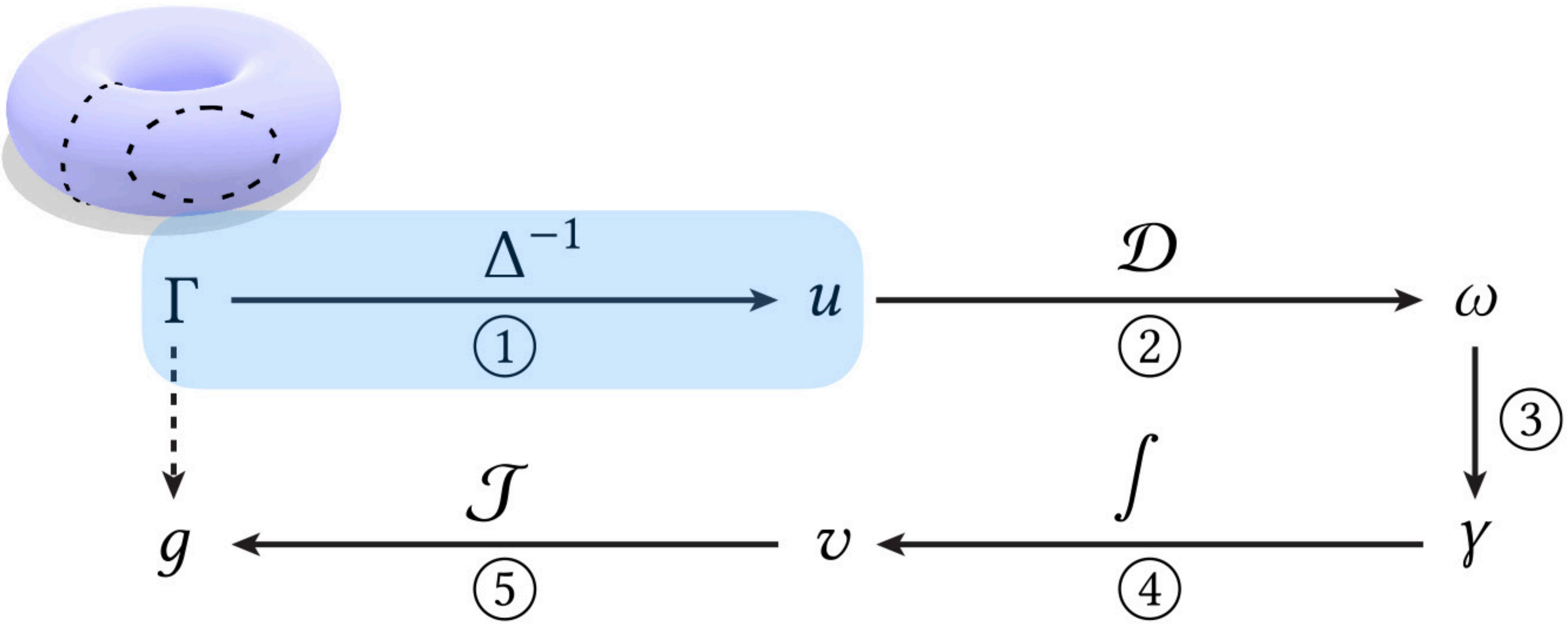
Singular point interpolation

$$f(\lambda_i, \lambda_j, \lambda_k) := \frac{\lambda_j f_j^{ki} + \lambda_k f_k^{ij}}{\lambda_j + \lambda_k}$$

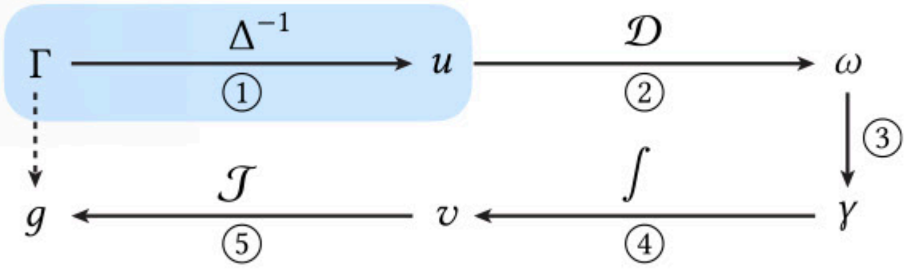




Solving the jump Laplace equation...



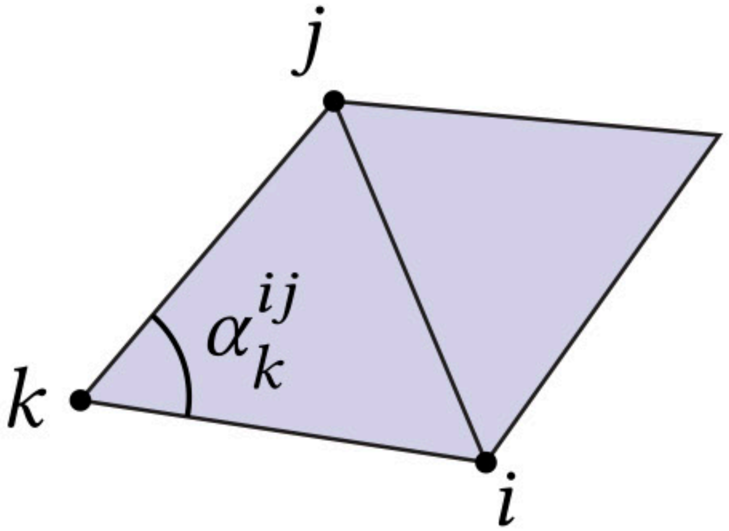
The discrete jump Laplacian



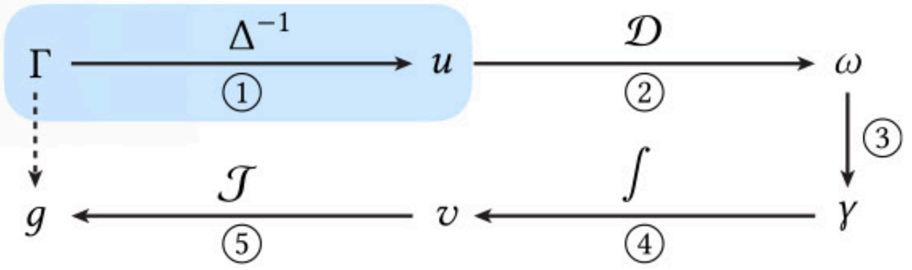
Build the standard cotan Laplacian on V^*:

$$
\begin{aligned}
L_{ij} &= L_{ji} = -\mathrm{w}_{ij}, && \forall ij \in E^* \\
L_{ii} &= \textstyle\sum_{\in E^*} \mathrm{w}_{ij}, && \forall i \in V^*
\end{aligned}
$$

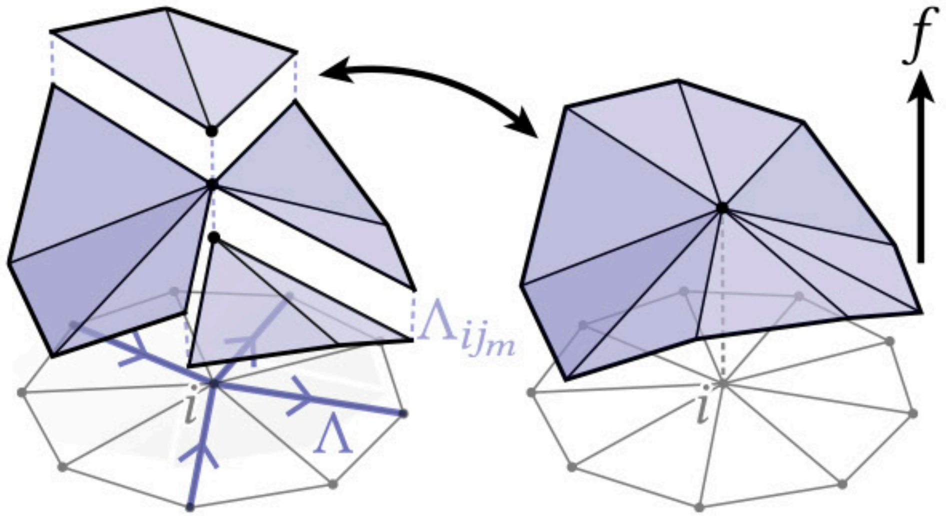
$$\mathrm{w}_{ij} := \frac{1}{2} \sum_{ijk \in F} \cot \alpha_k^{ij}$$



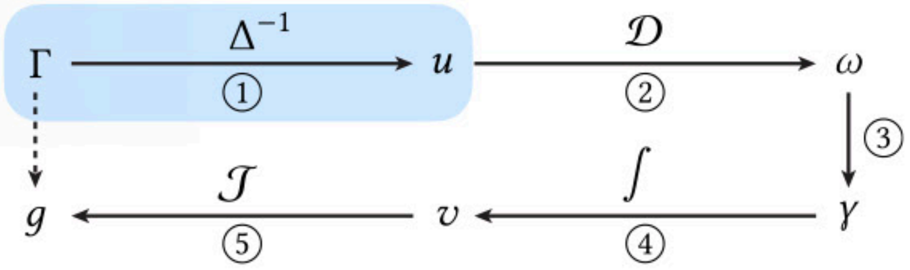
Reduced coordinates



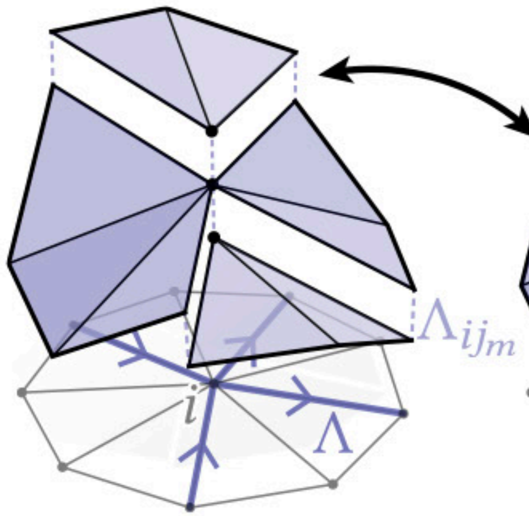
A jump harmonic function f is harmonic "up to jumps" Λ.



Reduced coordinates

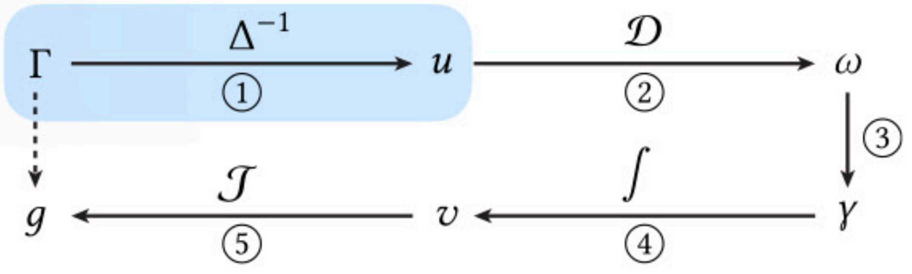


A jump harmonic function f is harmonic "up to jumps" Λ.

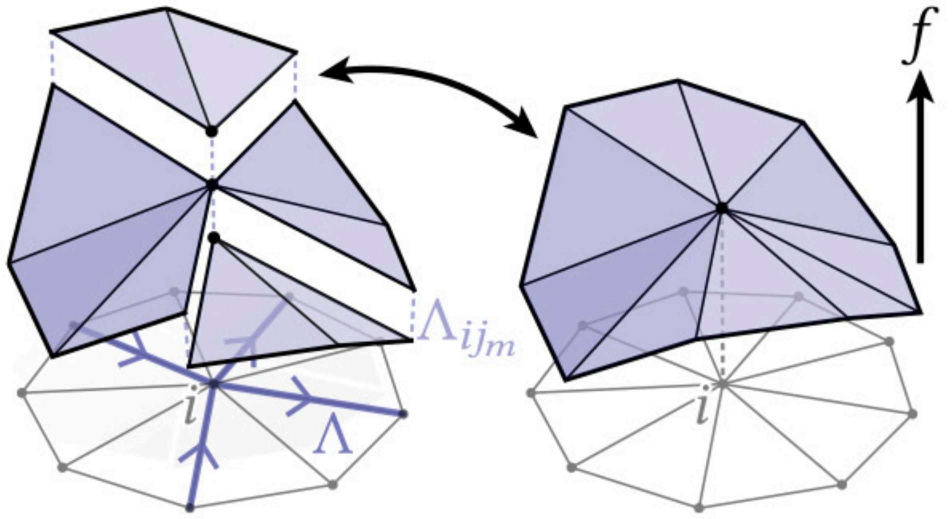


Value per corner →
one value per vertex!

Reduced coordinates



A jump harmonic function f is harmonic "up to jumps" Λ.

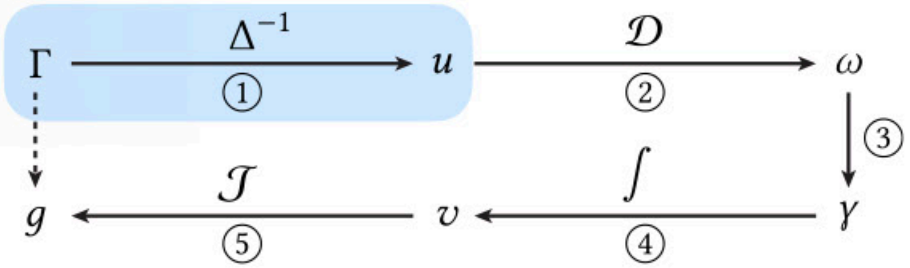


Value per corner →
one value per vertex!

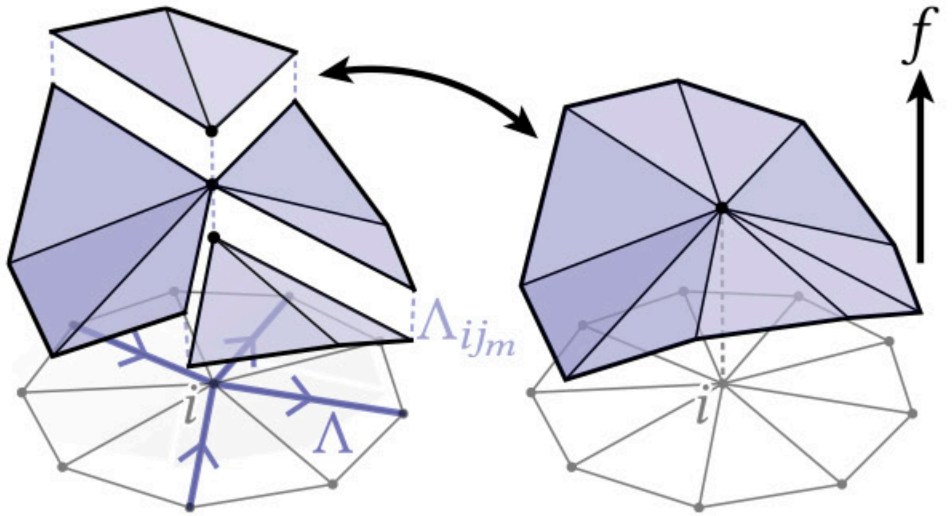
Apply change of variables and solve

$$Lf_0 = b$$

Reduced coordinates



A jump harmonic function f is harmonic "up to jumps" Λ.



Value per corner →
one value per vertex!

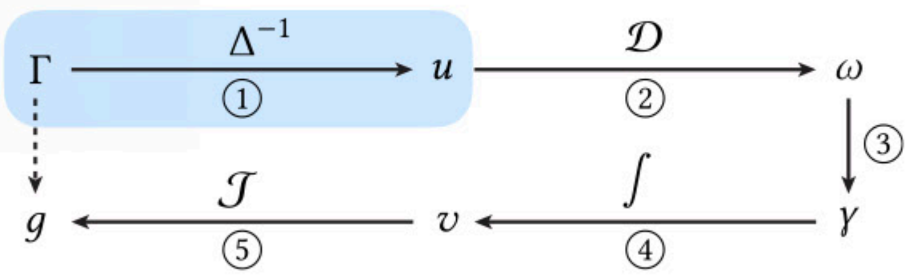
Apply change of variables and solve

$$Lf_0 = b$$

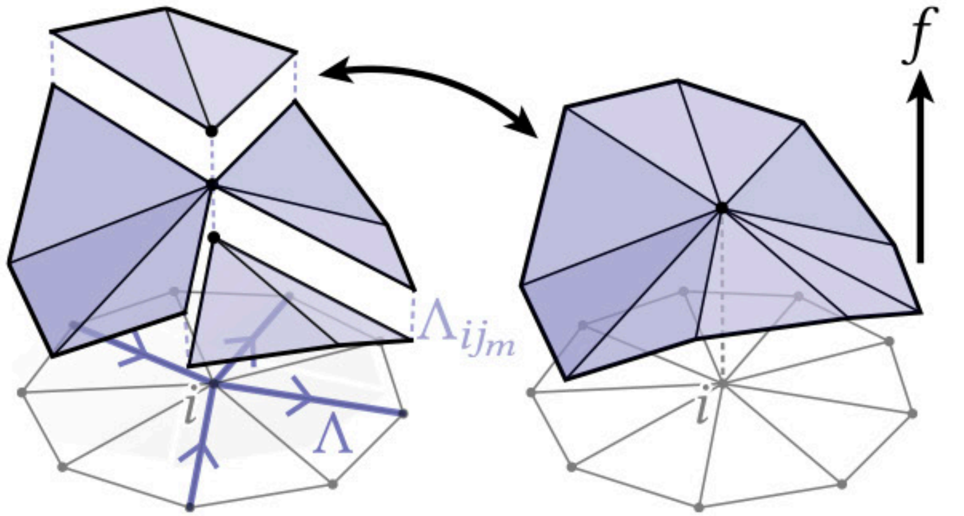
values per vertex

constant vector encoding per-corner jumps

Reduced coordinates



A jump harmonic function f is harmonic "up to jumps" Λ.



Value per corner →
one value per vertex!

Designing Quadrangulations with Discrete Harmonic Forms. Tong, Alliez, Cohen-Steiner, Desbrun (2006)

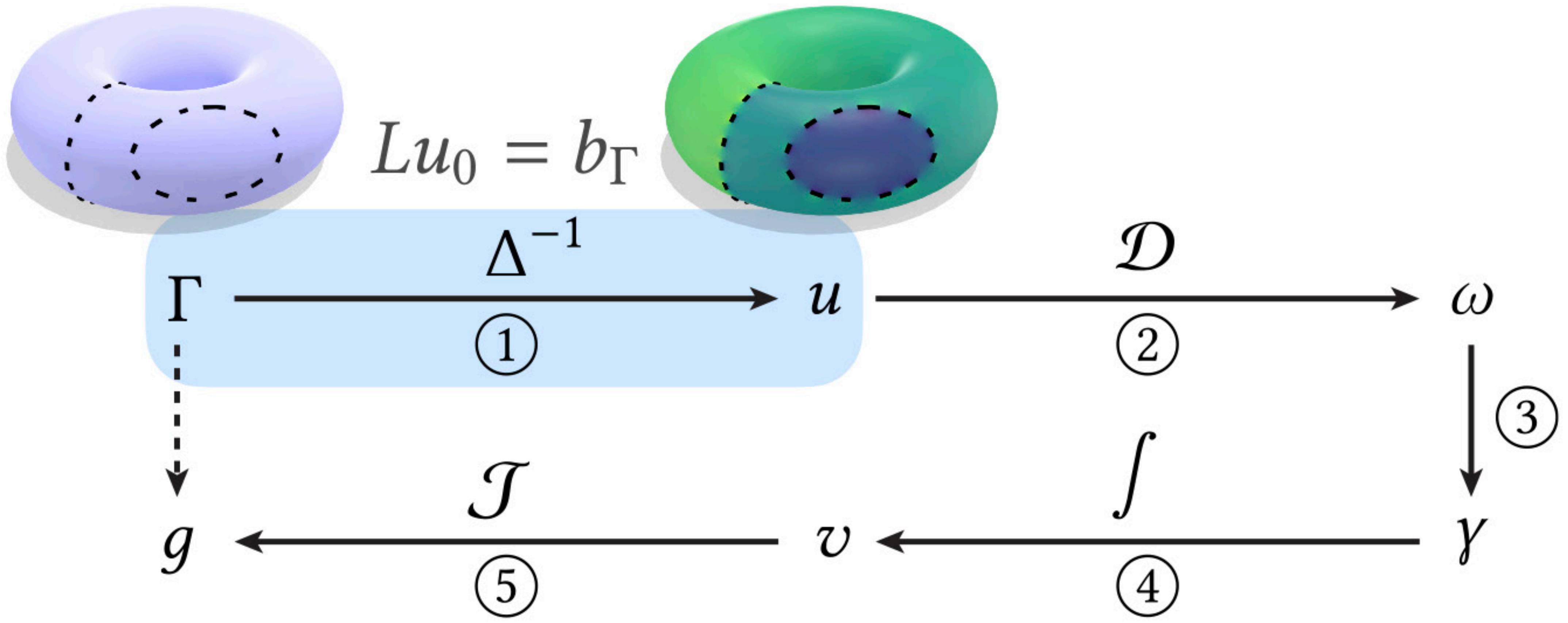
Apply change of variables and solve

$$L f_0 = b$$

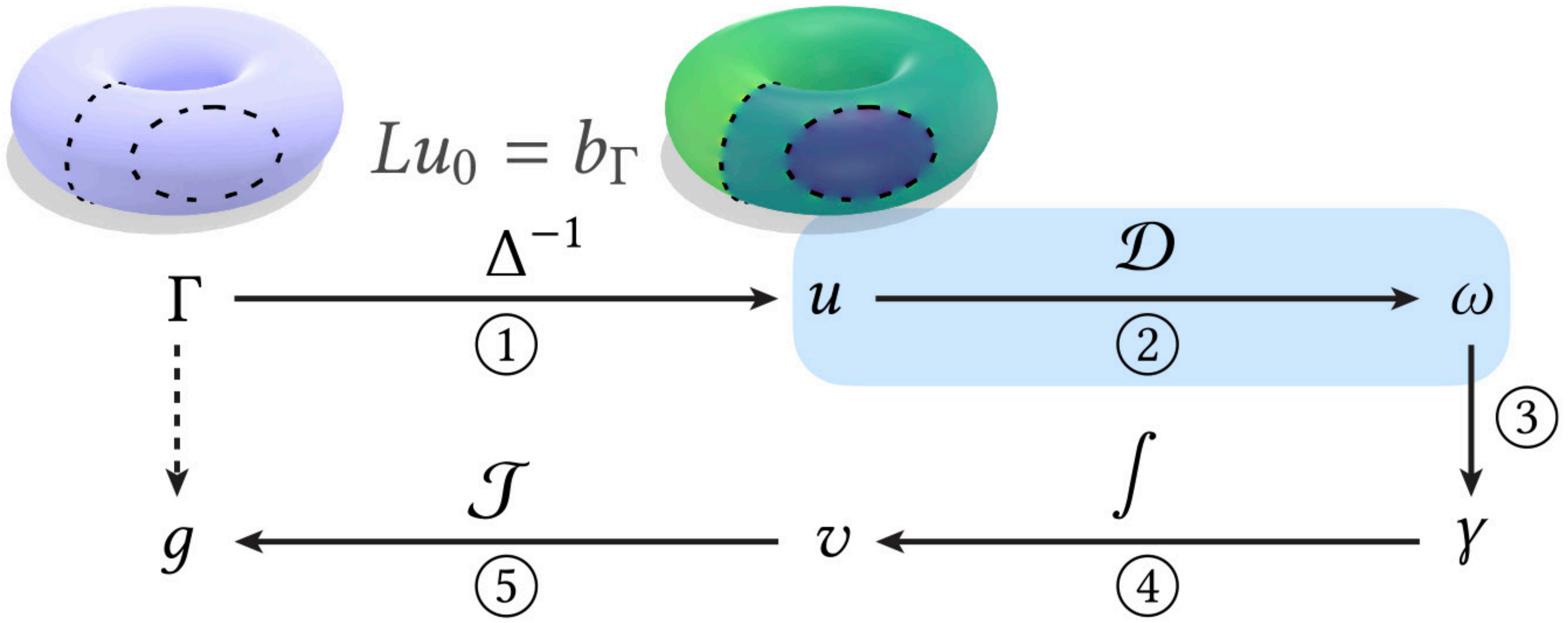
values per vertex

constant vector encoding per-corner jumps

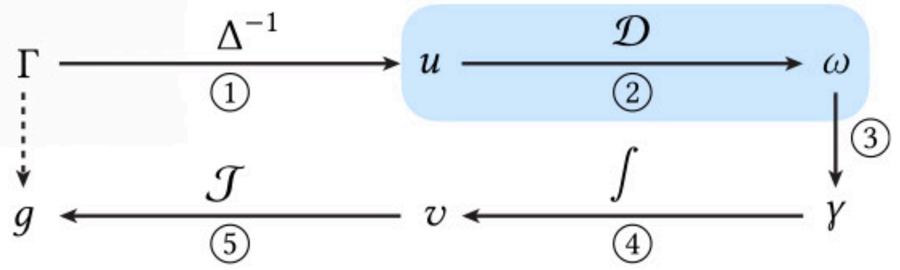
① Solve the jump Laplace equation for u

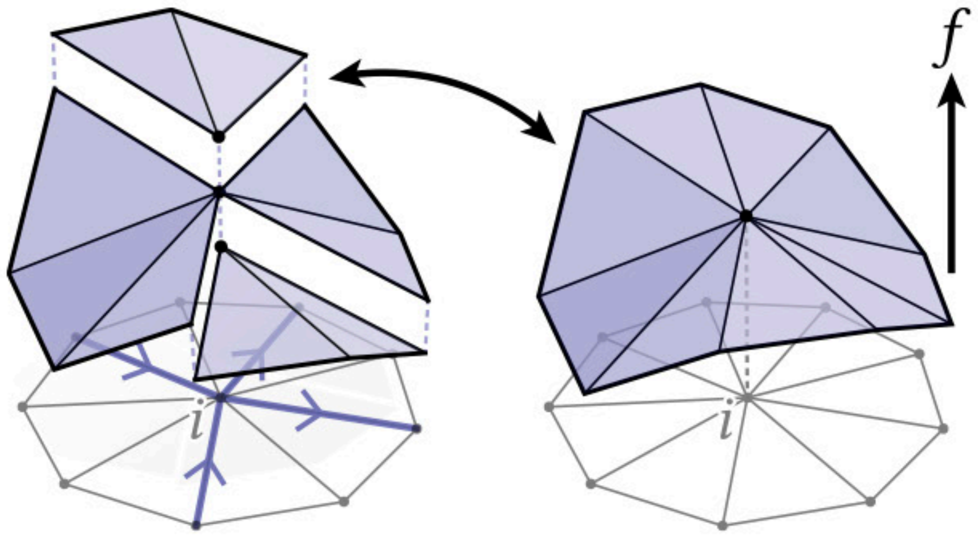


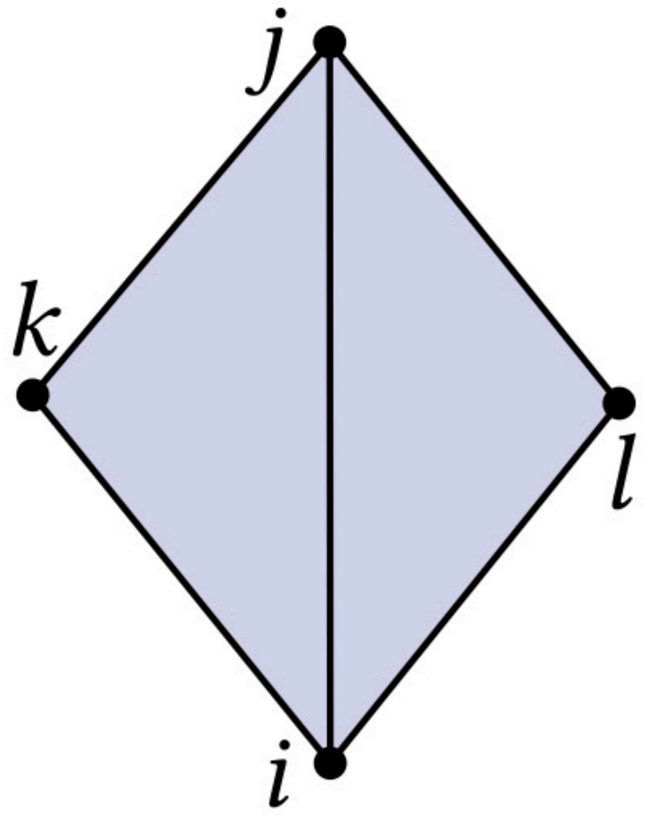
Jump harmonic functions → Derivatives...



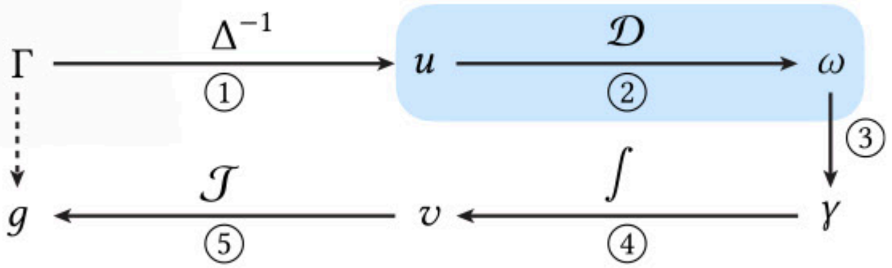
Differentiation

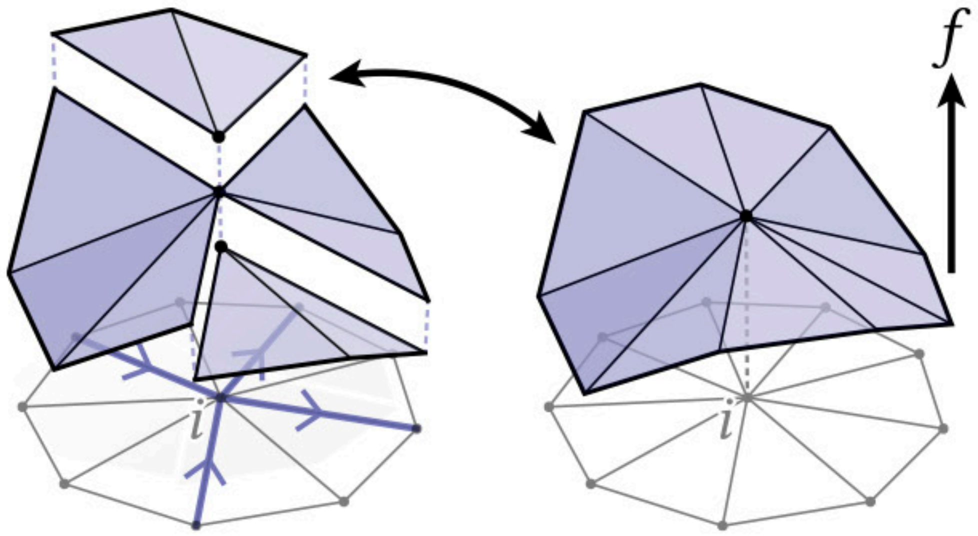


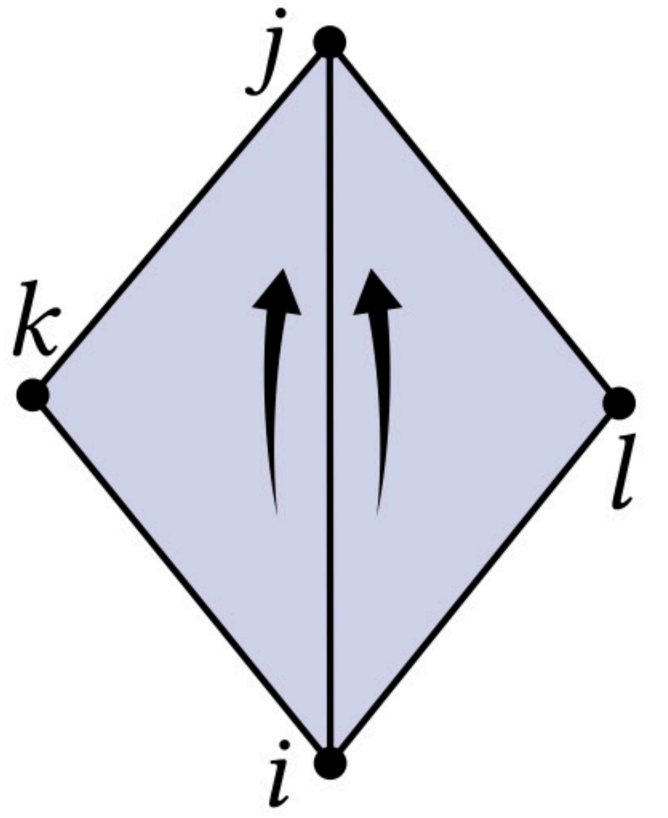




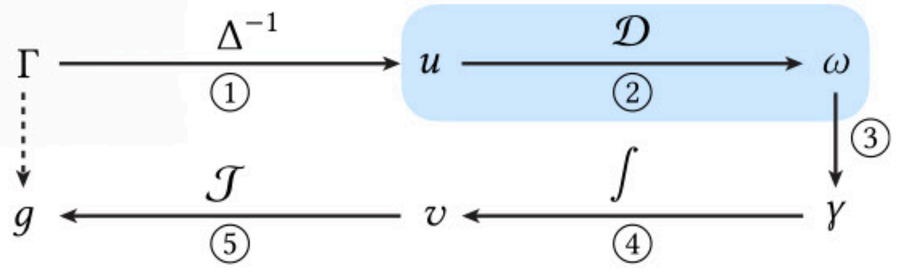
Differentiation

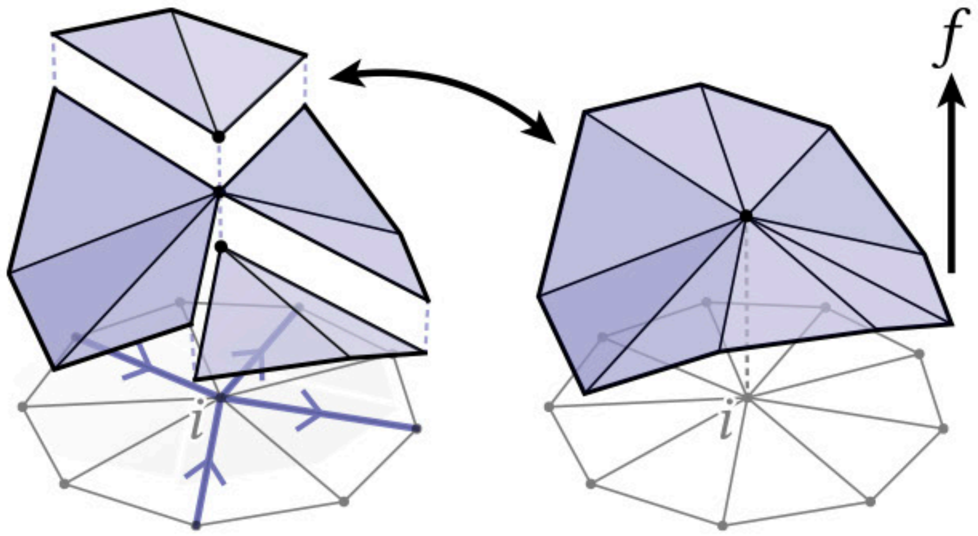


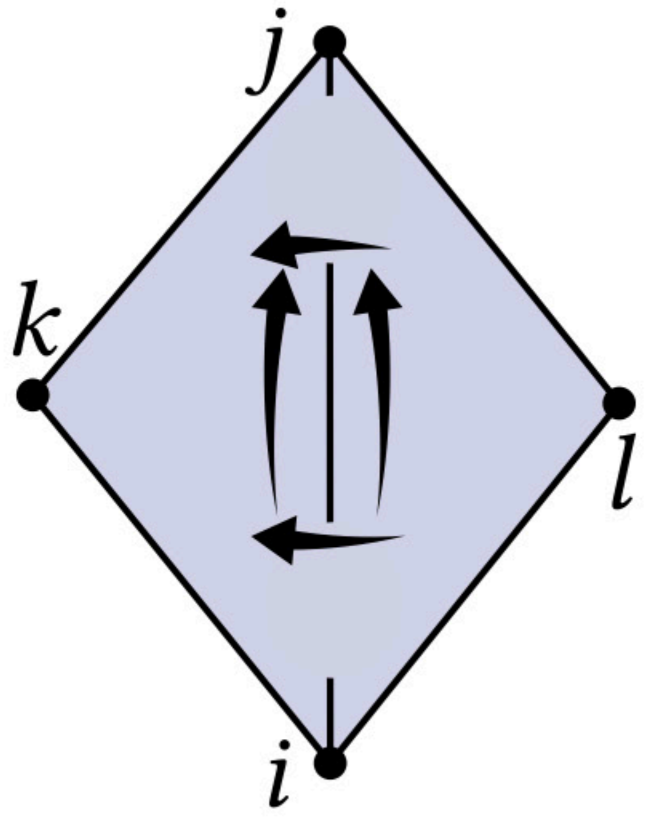




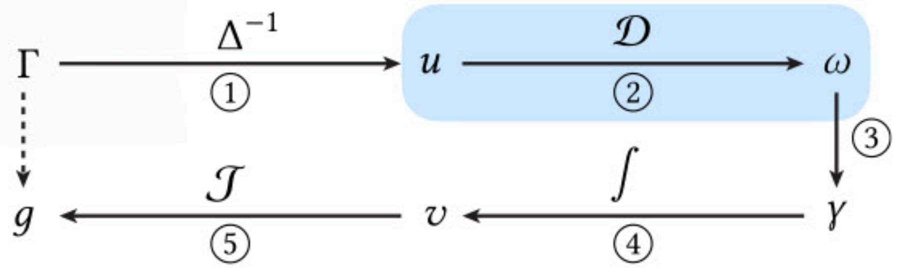
Differentiation

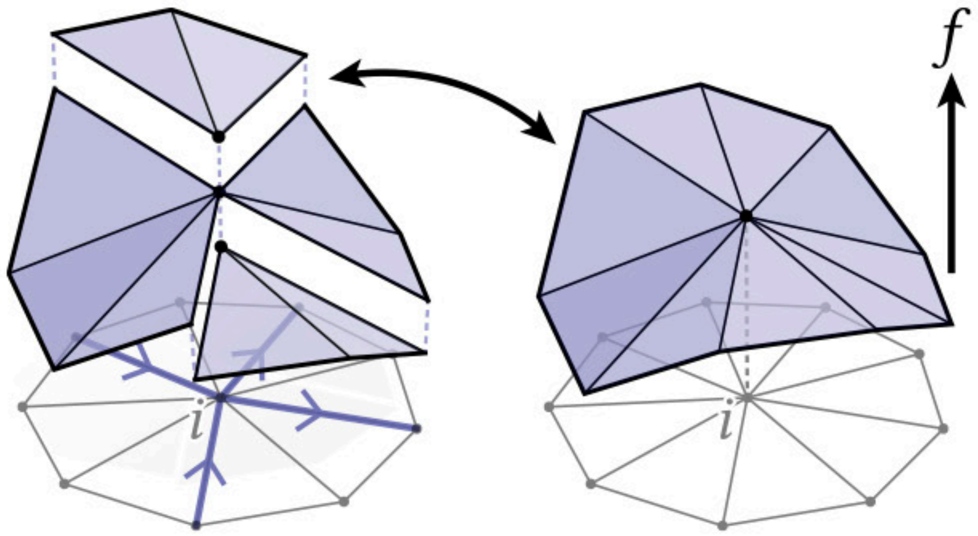






Differentiation

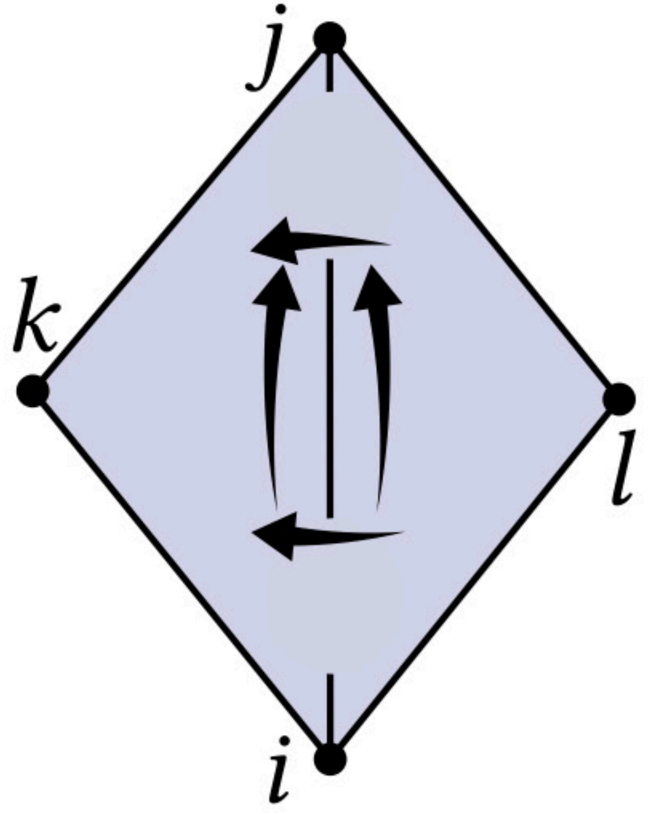




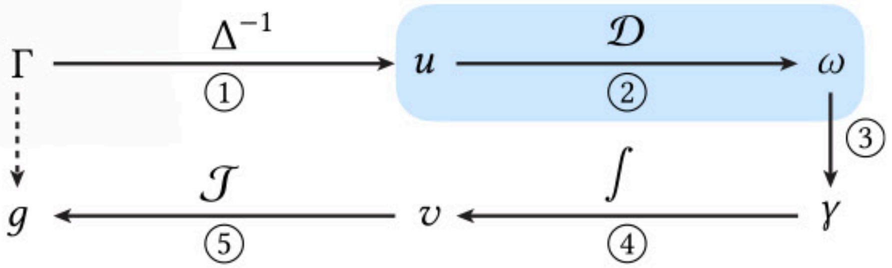
⇛

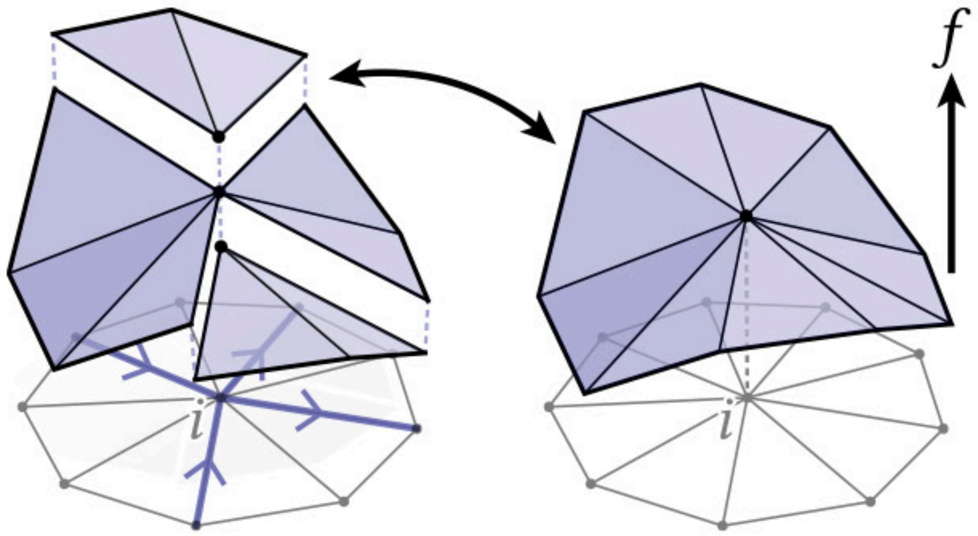
Jump compatibility condition:

$$f_i^{jk} - f_i^{lj} = f_j^{ki} - f_j^{il}$$



Differentiation

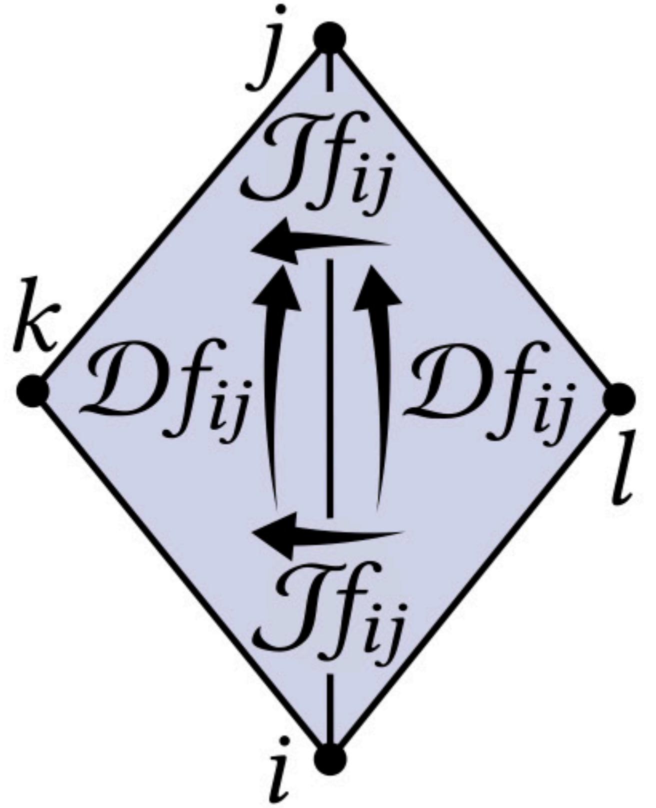




$\Rightarrow$

Jump compatibility condition:

$$f_i^{jk} - f_i^{lj} = f_j^{ki} - f_j^{il}$$

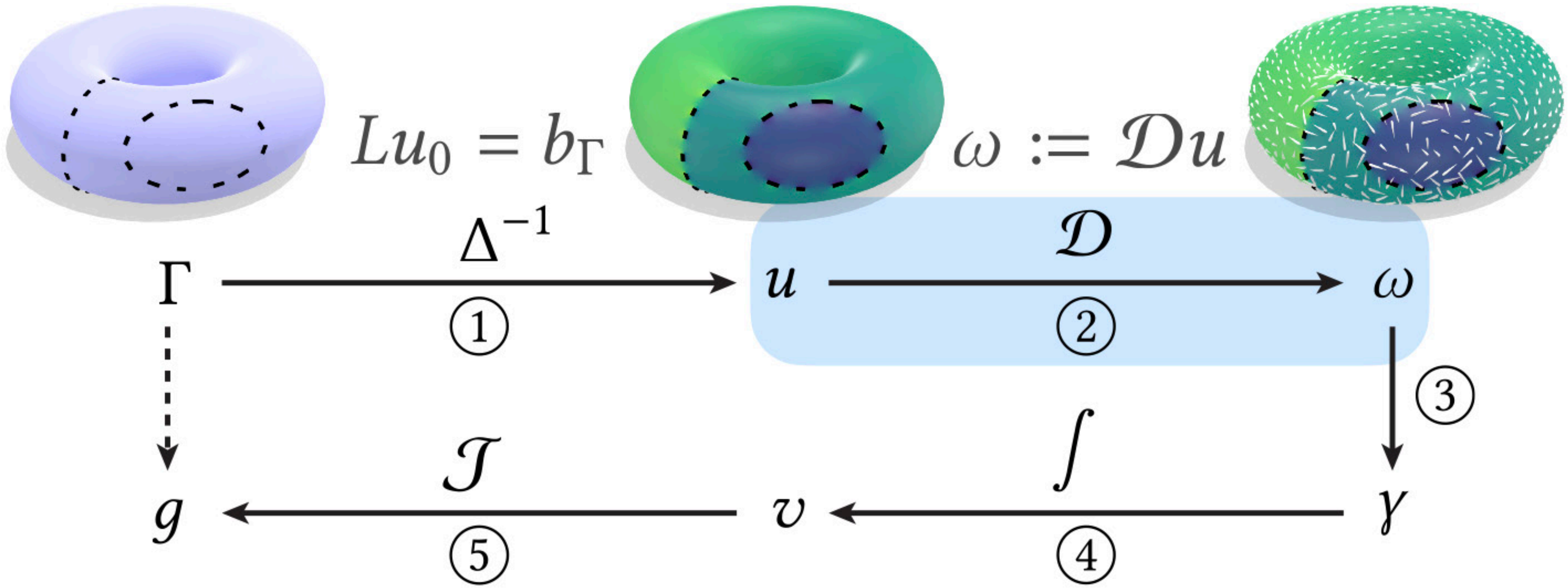


Darboux derivative: $(\mathcal{D}f)_{ij} := f_j^{ki} - f_i^{jk}$

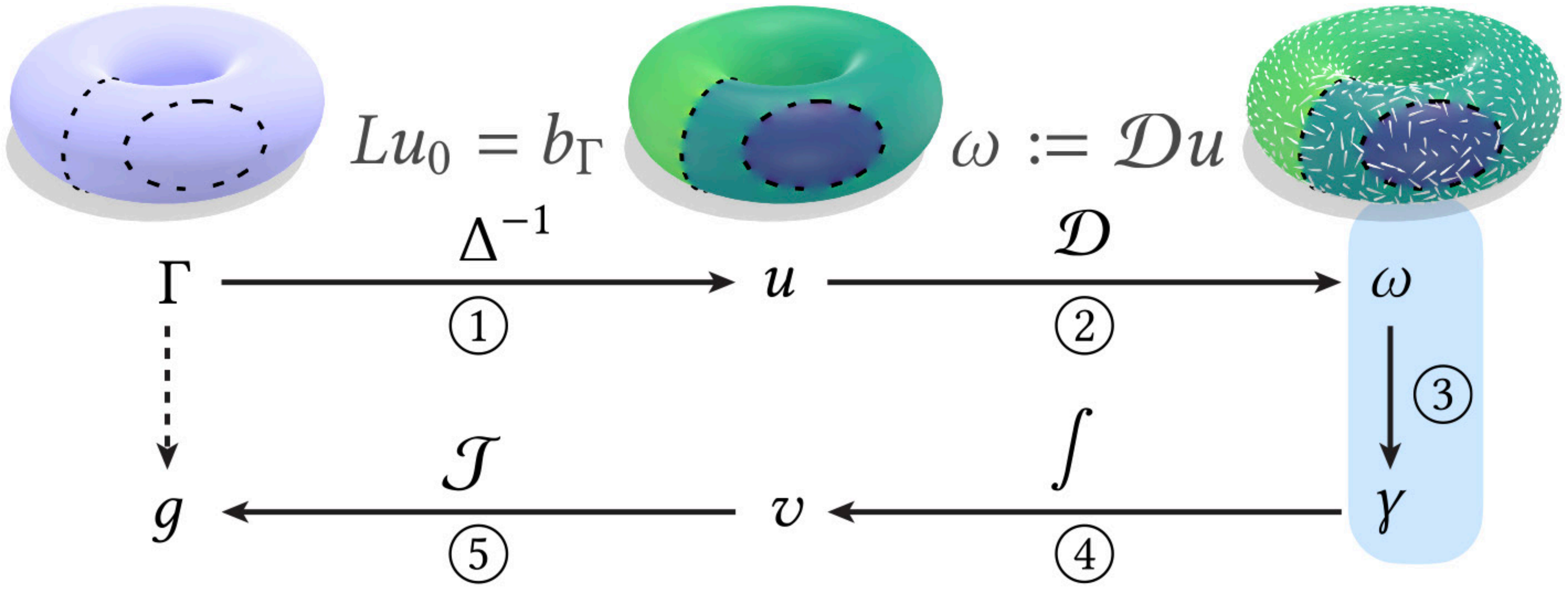
Jump derivative: $(\mathcal{J}f)_{ij} := f_i^{jk} - f_i^{lj}$

*subject to curve endpoints or nonmanifold edges

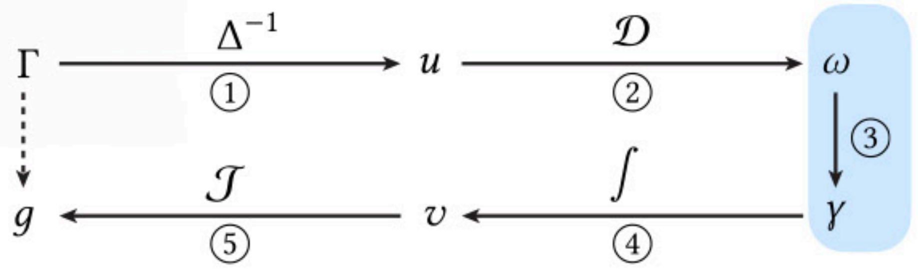
② Differentiate u



Derivative decomposition...

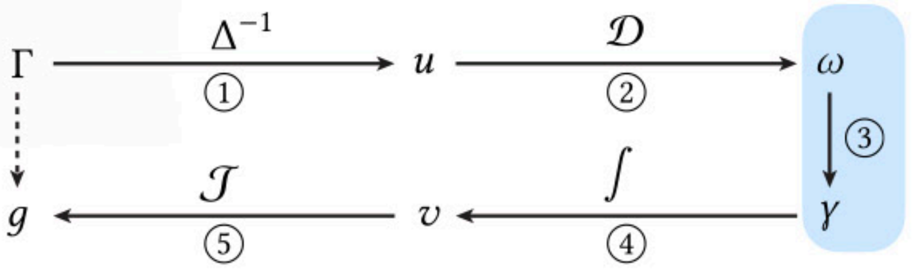


Hodge decomposition



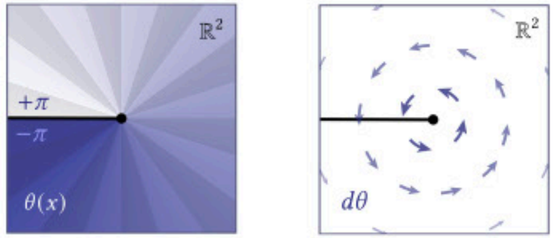
$$\omega = d\alpha + \delta\beta + \gamma$$

Hodge decomposition

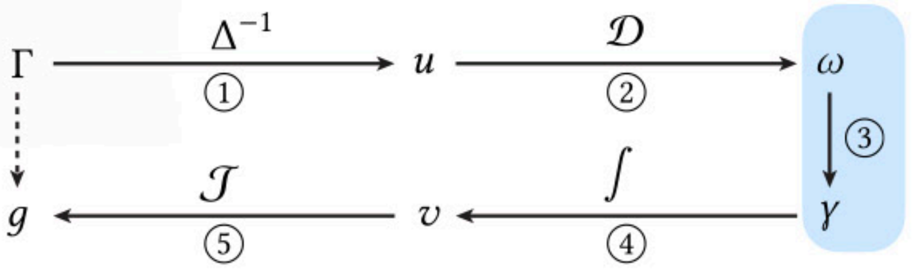


$$\omega = \cancelto{0}{d\alpha} + \delta\beta + \gamma$$

Lemma, Appendix A

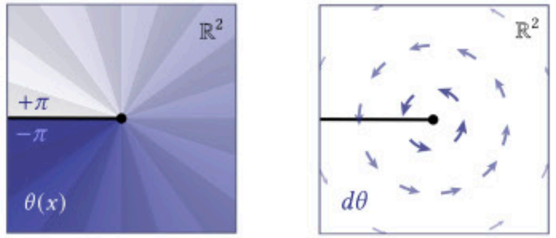


Hodge decomposition



$$\omega = \cancelto{0}{d\alpha} + \delta\beta + \gamma$$

Lemma, Appendix A

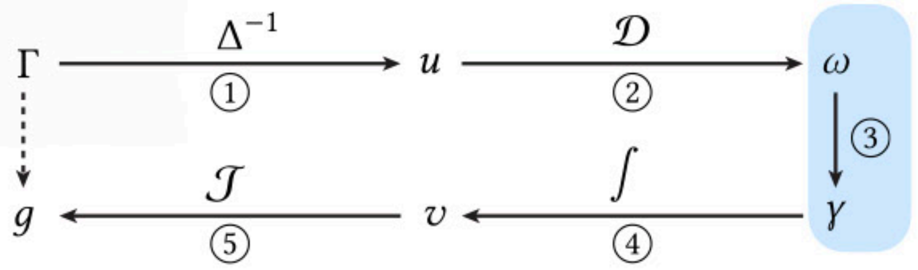


Solve Poisson equation: $\Delta_2 \beta = d_1 \omega$

$$\Delta_2 := d_1 *_1^{-1} d_1^T *_2$$

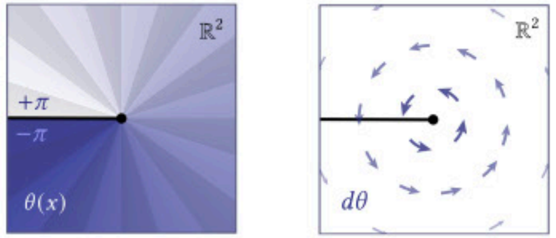
Discrete differential forms for computational modeling.
Desbrun, Kanso, Tong (2005)

Hodge decomposition



$$\omega = \cancel{d\alpha}^{\,0} + \delta\beta + \gamma$$

Lemma, Appendix A



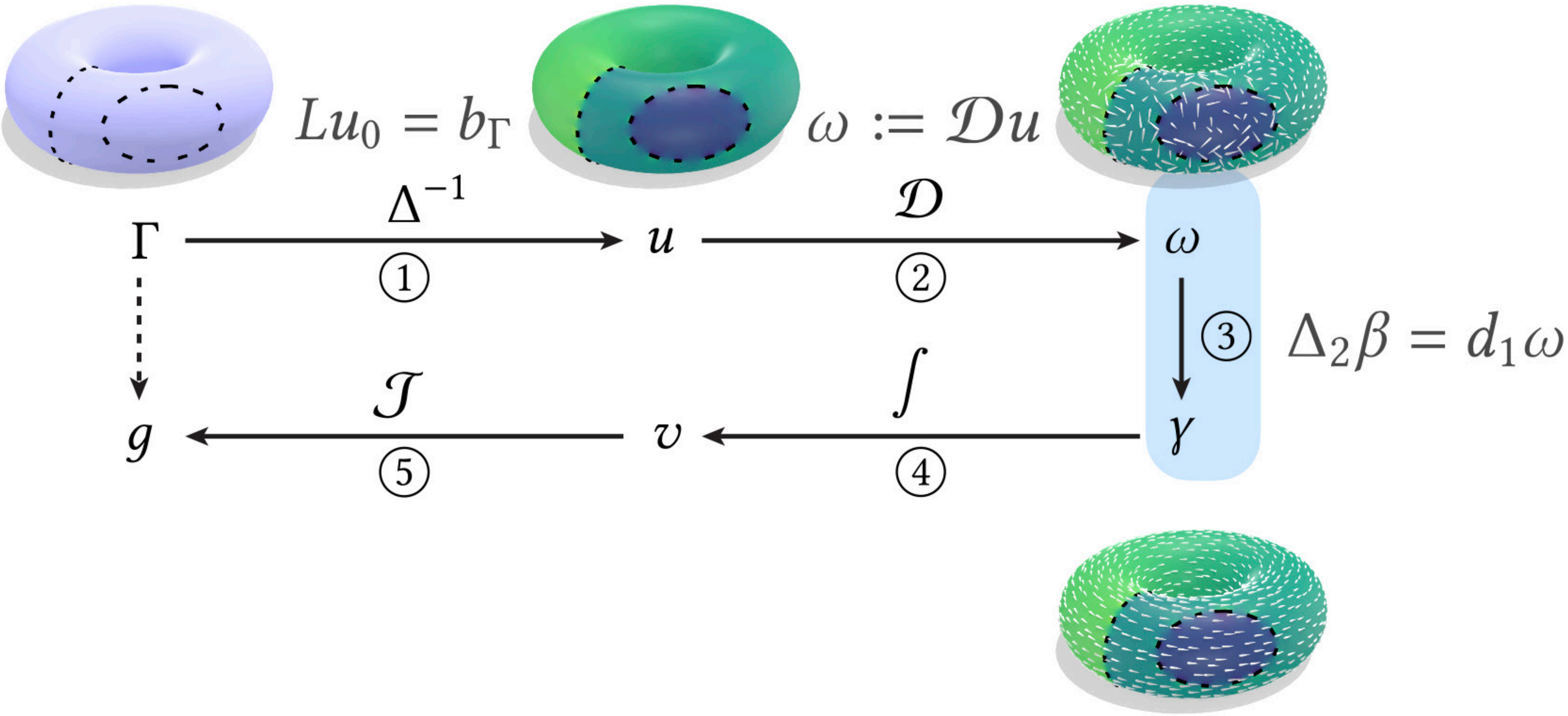
Solve Poisson equation: $\Delta_2 \beta = d_1 \omega$

$$\Delta_2 := d_1 *_1^{-1} d_1^T *_2$$

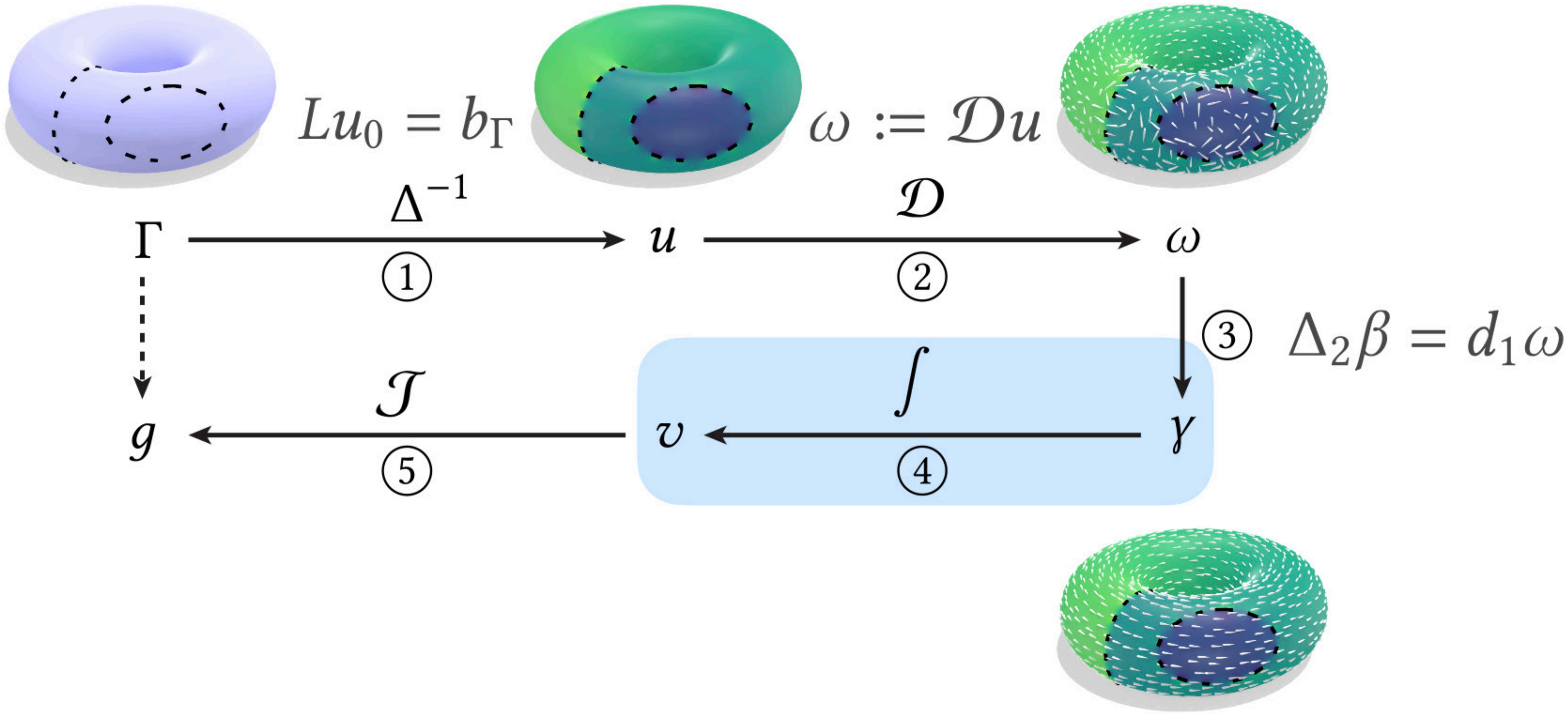
Get harmonic component: $\gamma \leftarrow \omega - \delta\beta$

Discrete differential forms for computational modeling.
Desbrun, Kanso, Tong (2005)

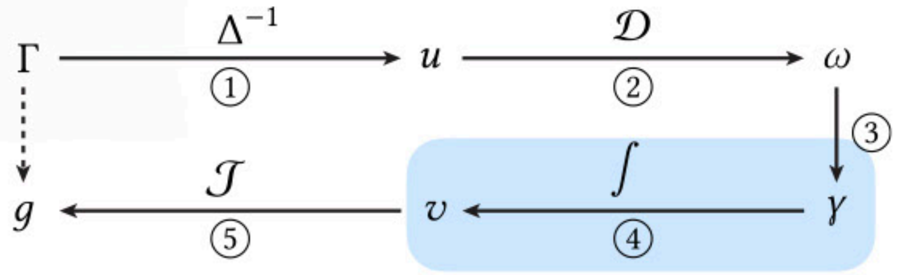
③ Hodge decompose ω



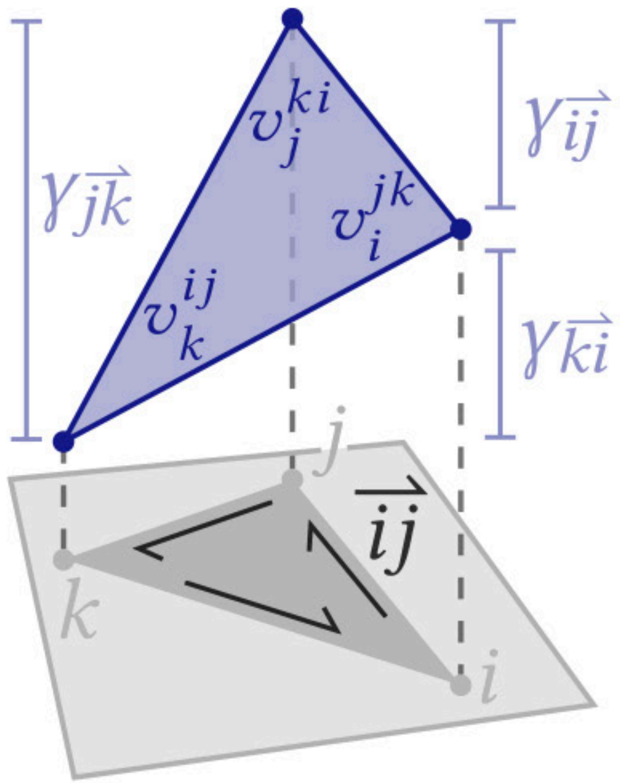
1-forms → Jump harmonic functions...



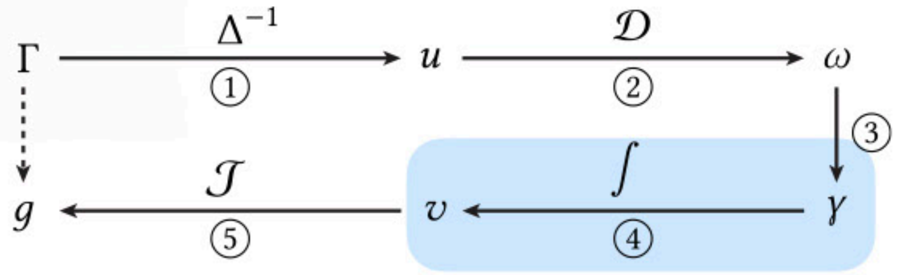
Integrating γ with jumps



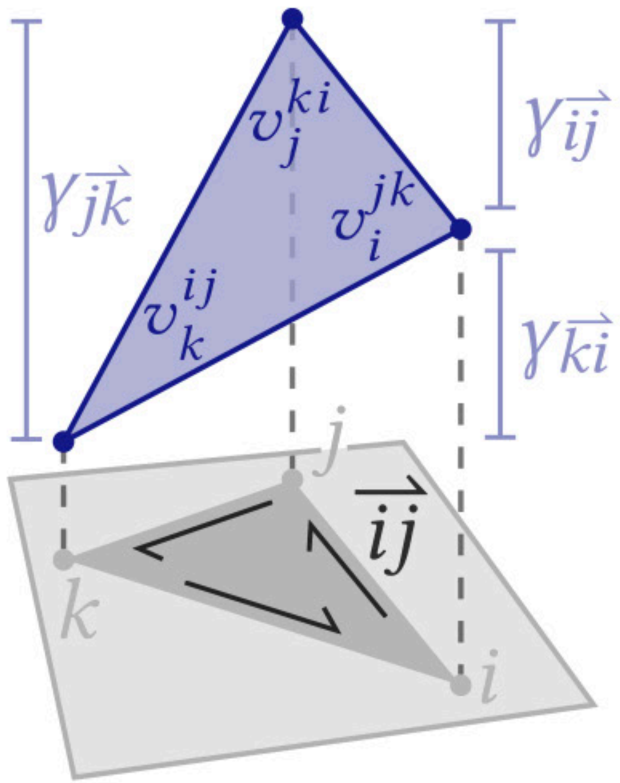
First locally integrate γ in each triangle.



Integrating γ with jumps

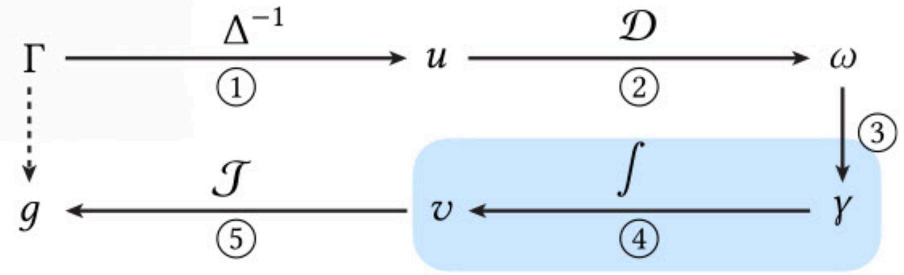


First locally integrate γ in each triangle.

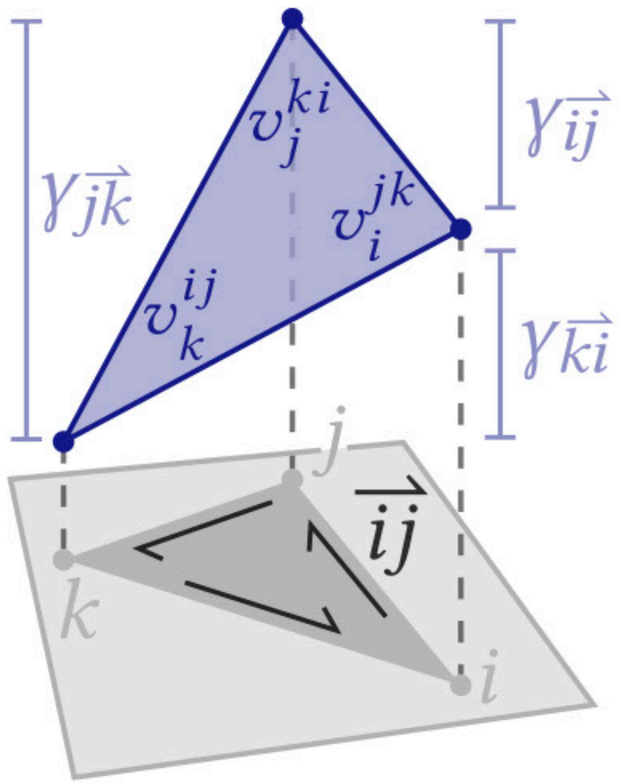


$$\mathring{v}_i^{jk} := 0$$

Integrating γ with jumps



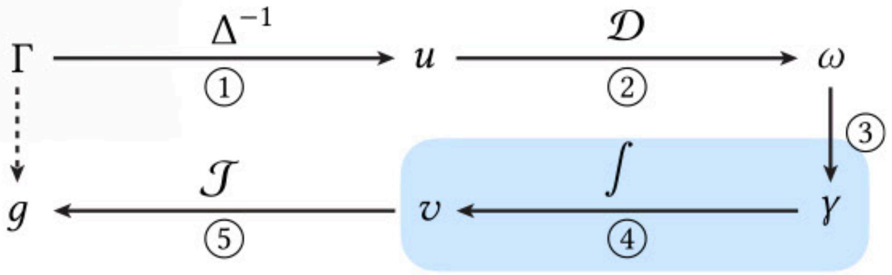
First locally integrate γ in each triangle.



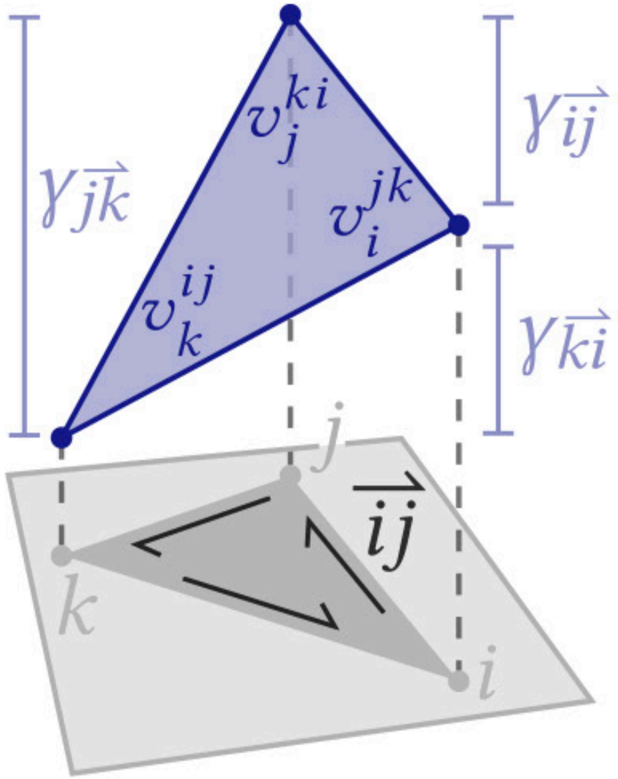
$$\mathring{v}_i^{jk} := 0$$

$$\mathring{v}_j^{ki} := \gamma_{\overrightarrow{ij}}$$

Integrating γ with jumps



First locally integrate γ in each triangle.

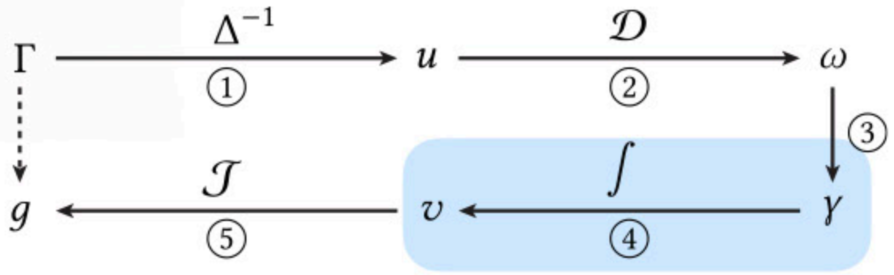


$$\mathring{v}_i^{jk} := 0$$

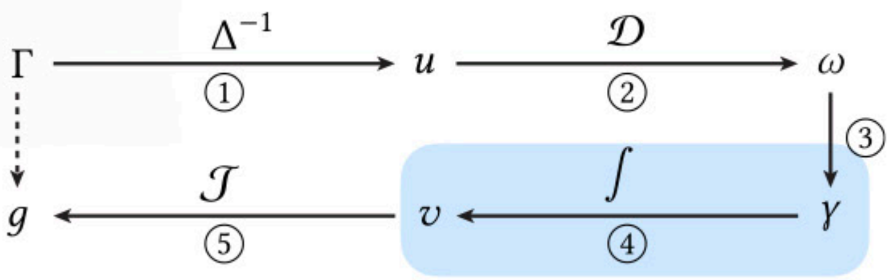
$$\mathring{v}_j^{ki} := \gamma_{\vec{ij}}$$

$$\mathring{v}_k^{ij} := \gamma_{\vec{ij}} + \gamma_{\vec{jk}}$$

Computing the residual function



Computing the residual function



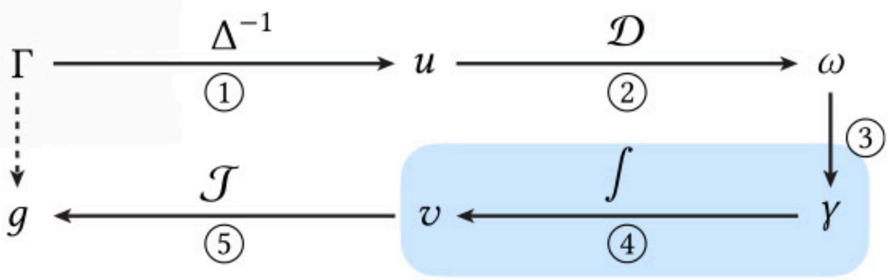
Applying a change of variable, optimize for $|F|$ per-face shifts:

$$\min_{\sigma \in \mathbb{R}^{|F|}} \sum_{ij \in \Gamma \cap E_{\text{int}}} \ell_{ij} |(\sigma_{ijk} - \sigma_{jil}) - s_{ij}| + \frac{1}{\varepsilon} \sum_{ij \in E_{\text{int}} \setminus \Gamma} \ell_{ij} |(\sigma_{ijk} - \sigma_{jil}) - s_{ij}|$$

$$\text{s.t.} \quad 0 \leq \frac{(\sigma_{ijk} - \sigma_{jil}) - s_{ij}}{\Gamma_{ij}} \leq 1, \quad \forall ij \in \Gamma.$$

linear program

Computing the residual function



Applying a change of variable, optimize for $|F|$ per-face shifts:

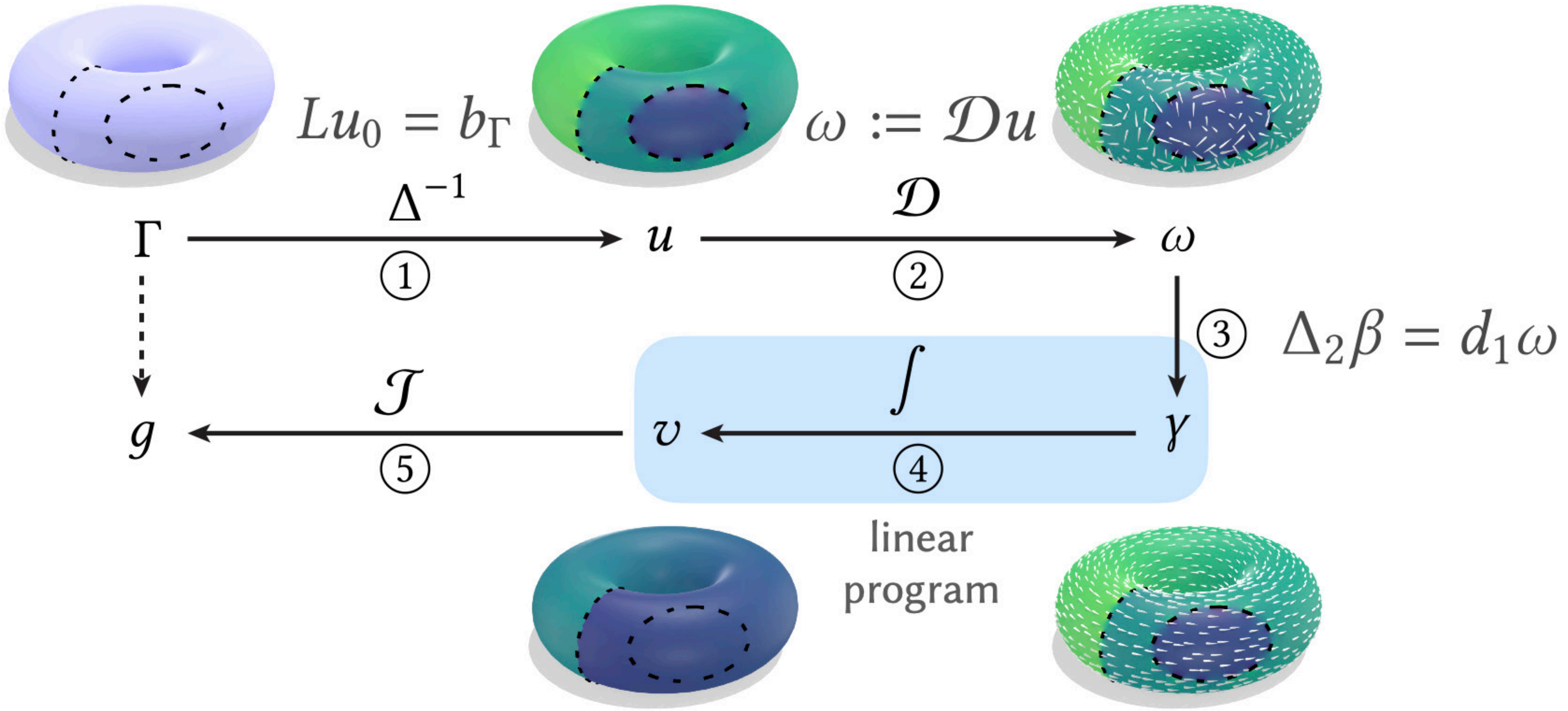
$$\min_{\sigma \in \mathbb{R}^{|F|}} \sum_{ij \in \Gamma \cap E_{\text{int}}} \ell_{ij} \left| (\sigma_{ijk} - \sigma_{jil}) - s_{ij} \right| + \frac{1}{\varepsilon} \sum_{ij \in E_{\text{int}} \setminus \Gamma} \ell_{ij} \left| (\sigma_{ijk} - \sigma_{jil}) - s_{ij} \right|$$

$$\text{s.t.} \quad 0 \leq \frac{(\sigma_{ijk} - \sigma_{jil}) - s_{ij}}{\Gamma_{ij}} \leq 1, \quad \forall ij \in \Gamma.$$

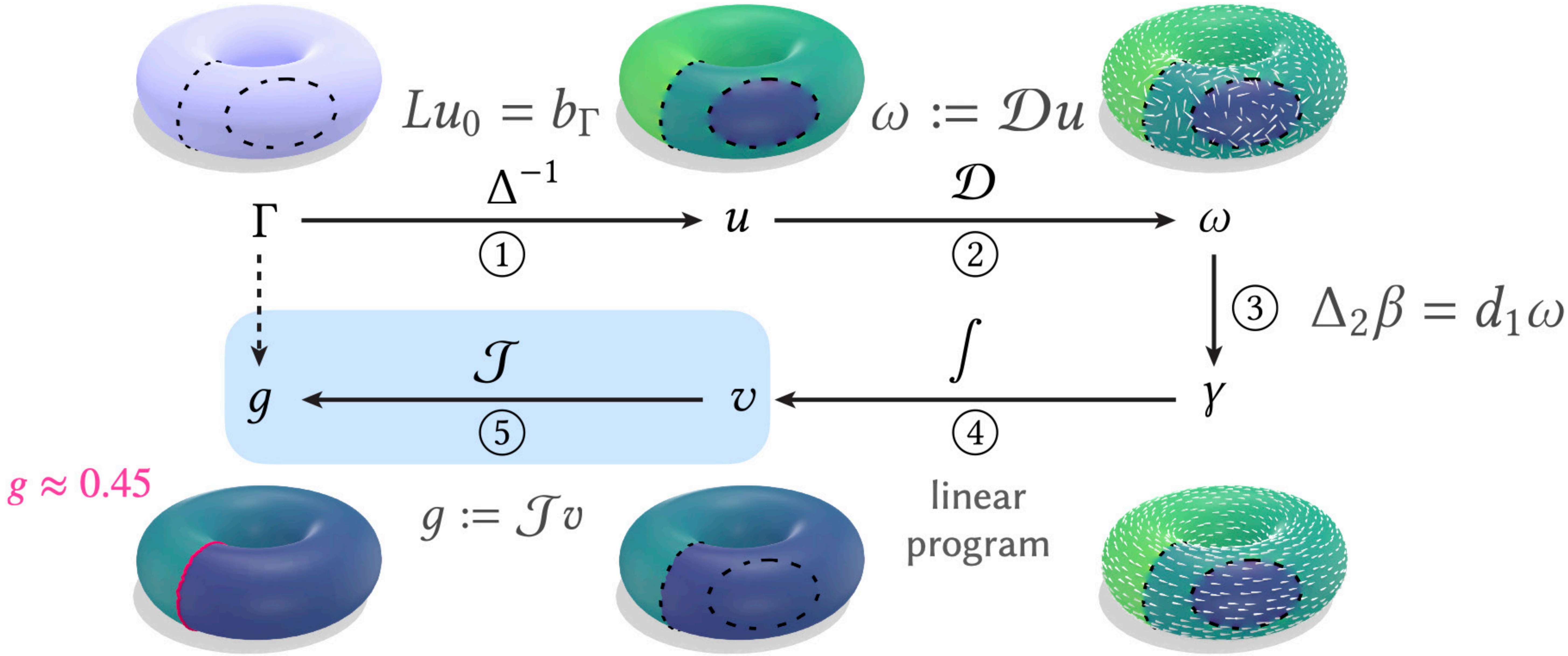
linear program

Afterwards, recover solution v.

④ Compute the residual function



⑤ Decompose curve



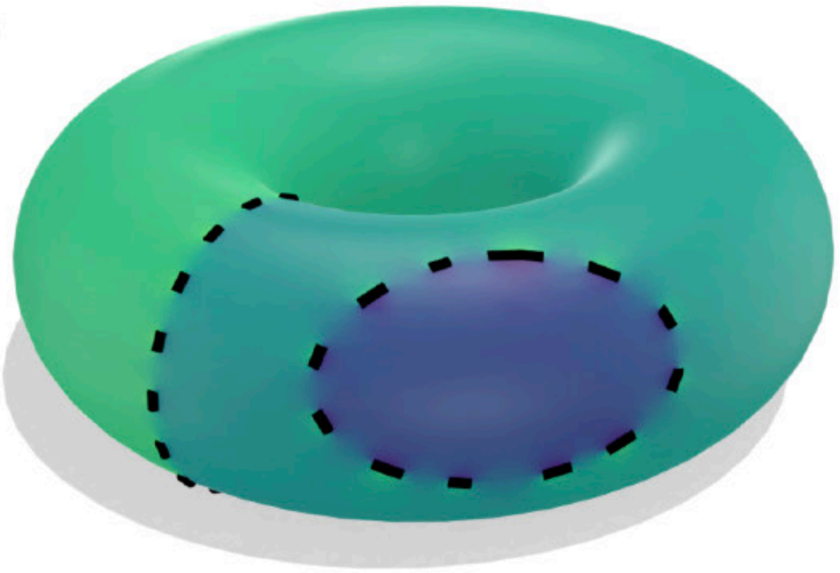
Winding number function

Solve jump Laplace equation with jumps $\Gamma - \mathcal{J}v$ to obtain w.

Winding number function

Solve jump Laplace equation with jumps $\Gamma - \mathcal{J}v$ to obtain w.

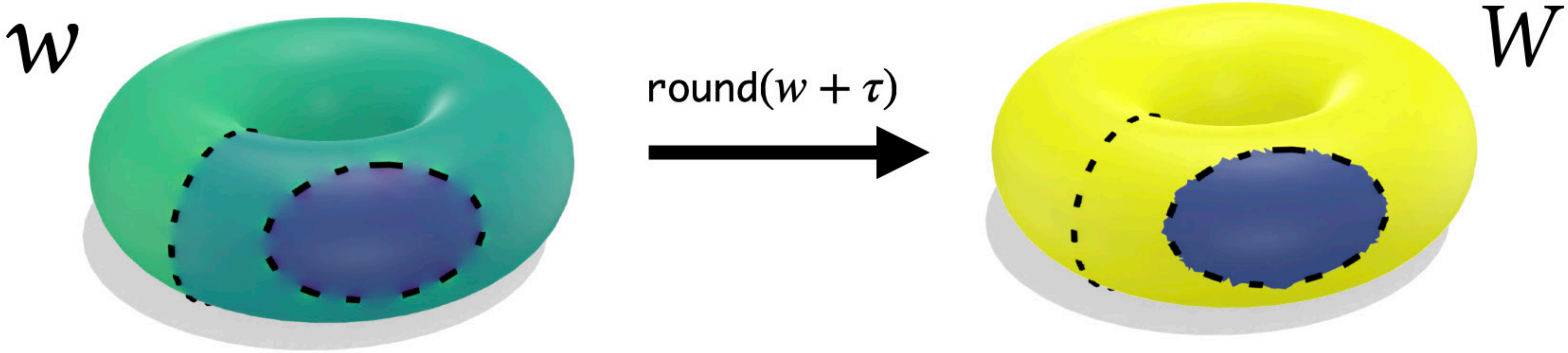
w



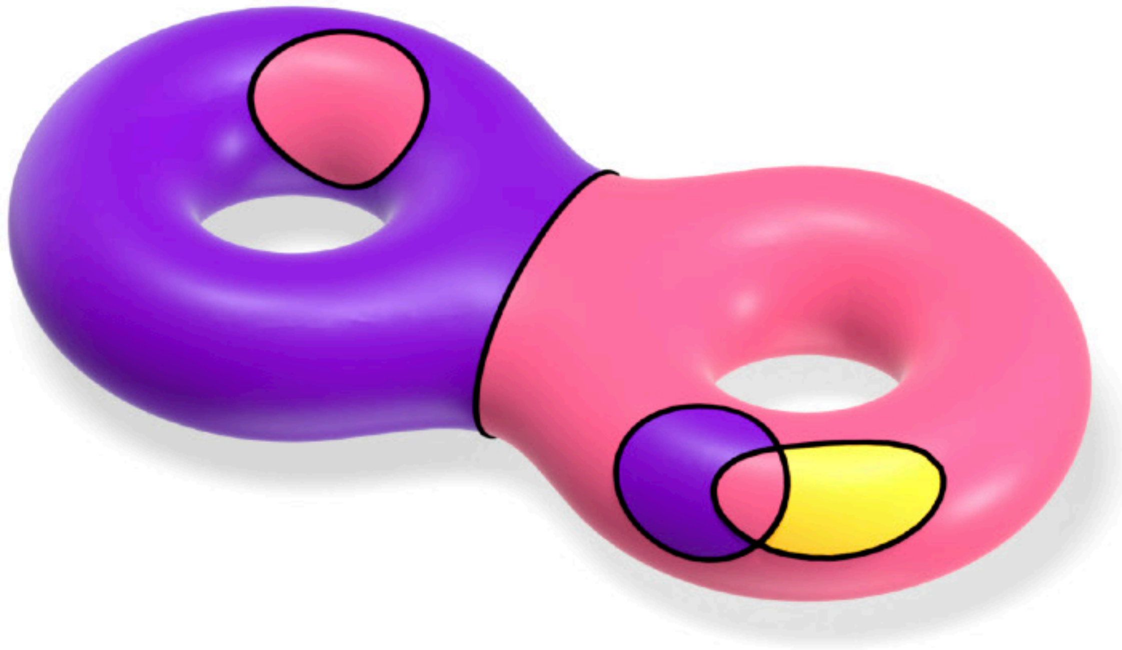
Winding number function

Solve jump Laplace equation with jumps $\Gamma - \mathcal{J} v$ to obtain w.

We compute a global shift τ such that $w + \tau$ is integer along Γ.

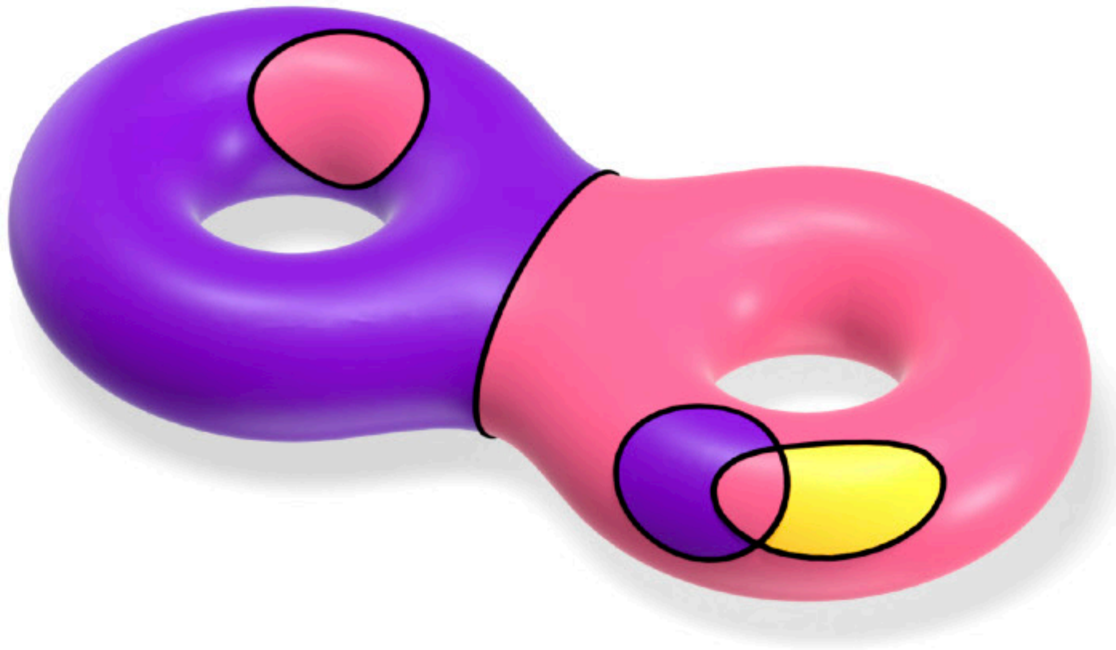


Summary



Summary

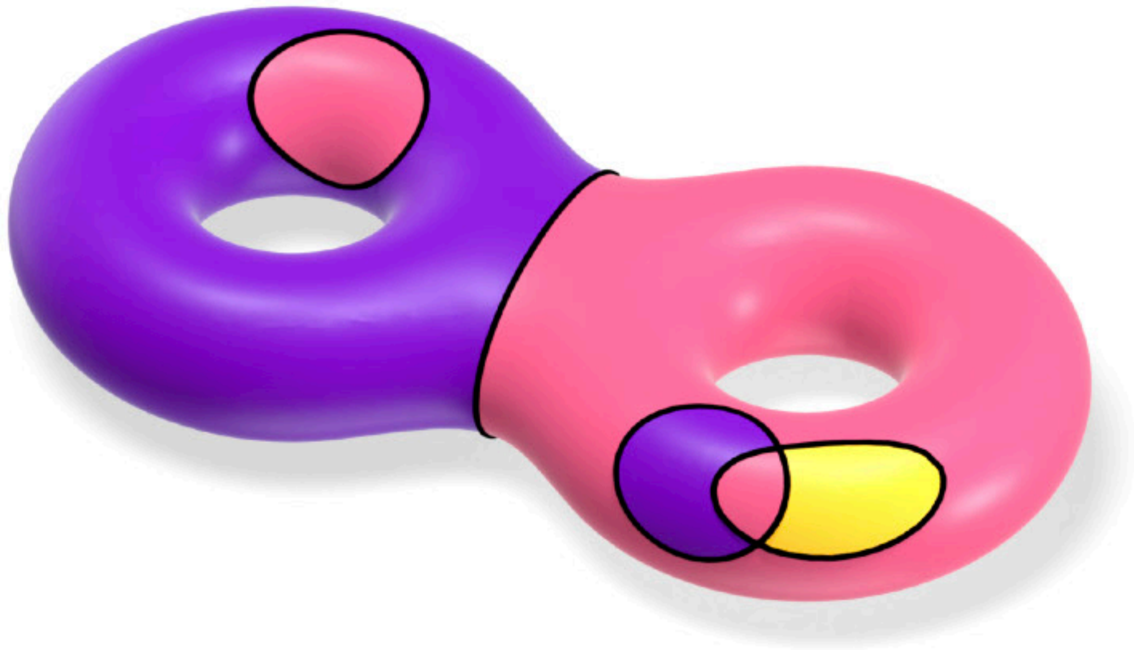
It's difficult to reason about the homology class of *broken* curves.



Summary

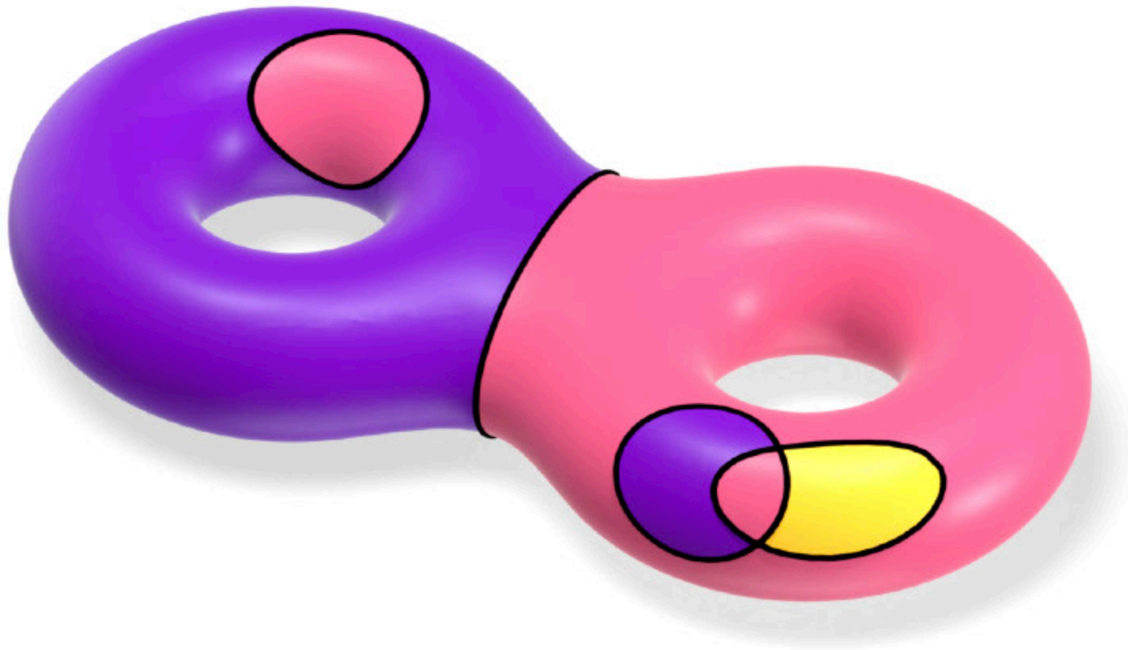
It's difficult to reason about the homology class of *broken* curves.

Instead of processing curves directly, we process functions *dual* to curves using *de Rham cohomology.*



Summary

It's difficult to reason about the homology class of *broken* curves.



Instead of processing curves directly, we process functions *dual* to curves using *de Rham cohomology.*

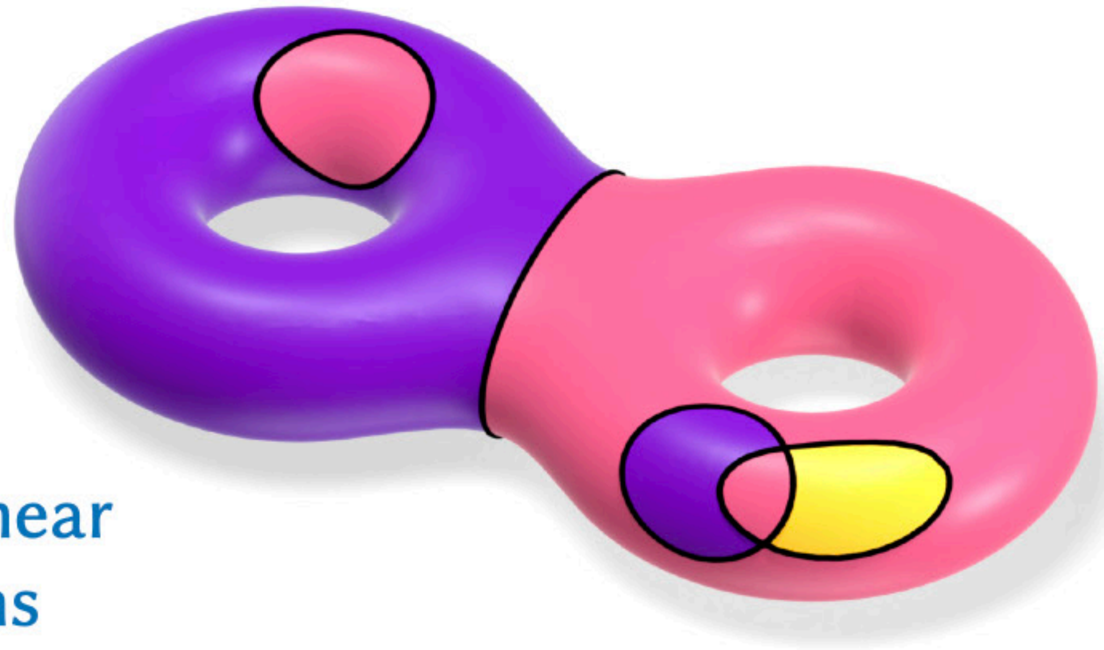
We map from functions back to curves, yielding integer region labels & identification of nonbounding components.

Summary

It's difficult to reason about the homology class of *broken* curves.

Instead of processing curves directly, we process functions *dual* to curves using *de Rham cohomology.*

sparse linear systems



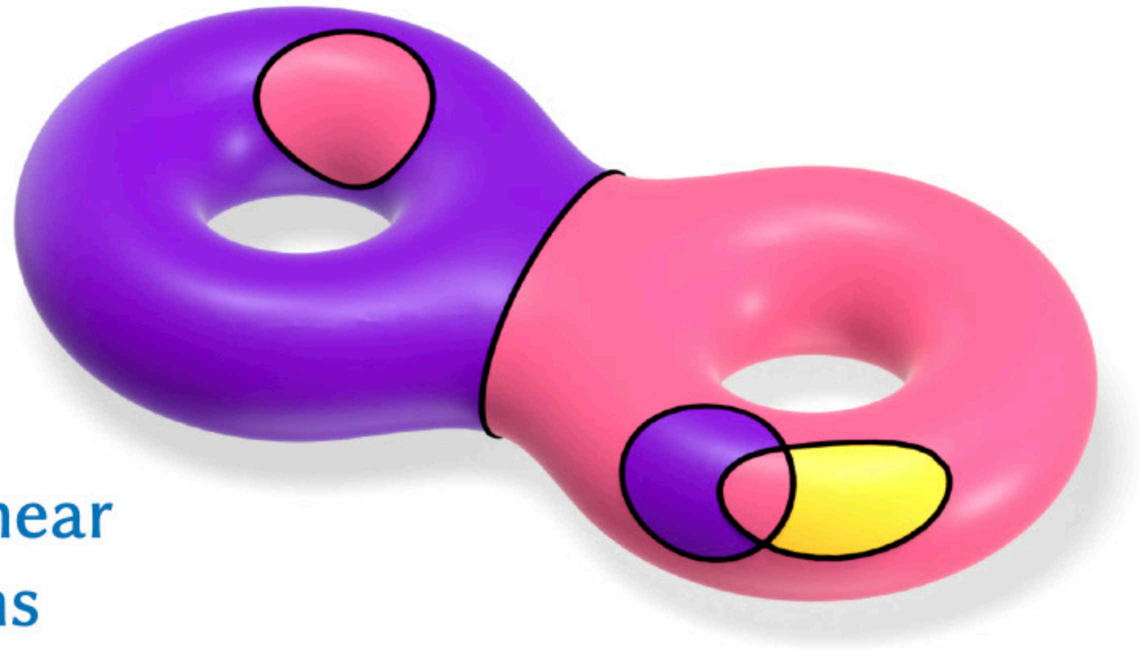
We map from functions back to curves, yielding integer region labels & identification of nonbounding components.

Summary

It's difficult to reason about the homology class of *broken* curves.

Instead of processing curves directly, we process functions *dual* to curves using *de Rham cohomology.*

sparse linear systems

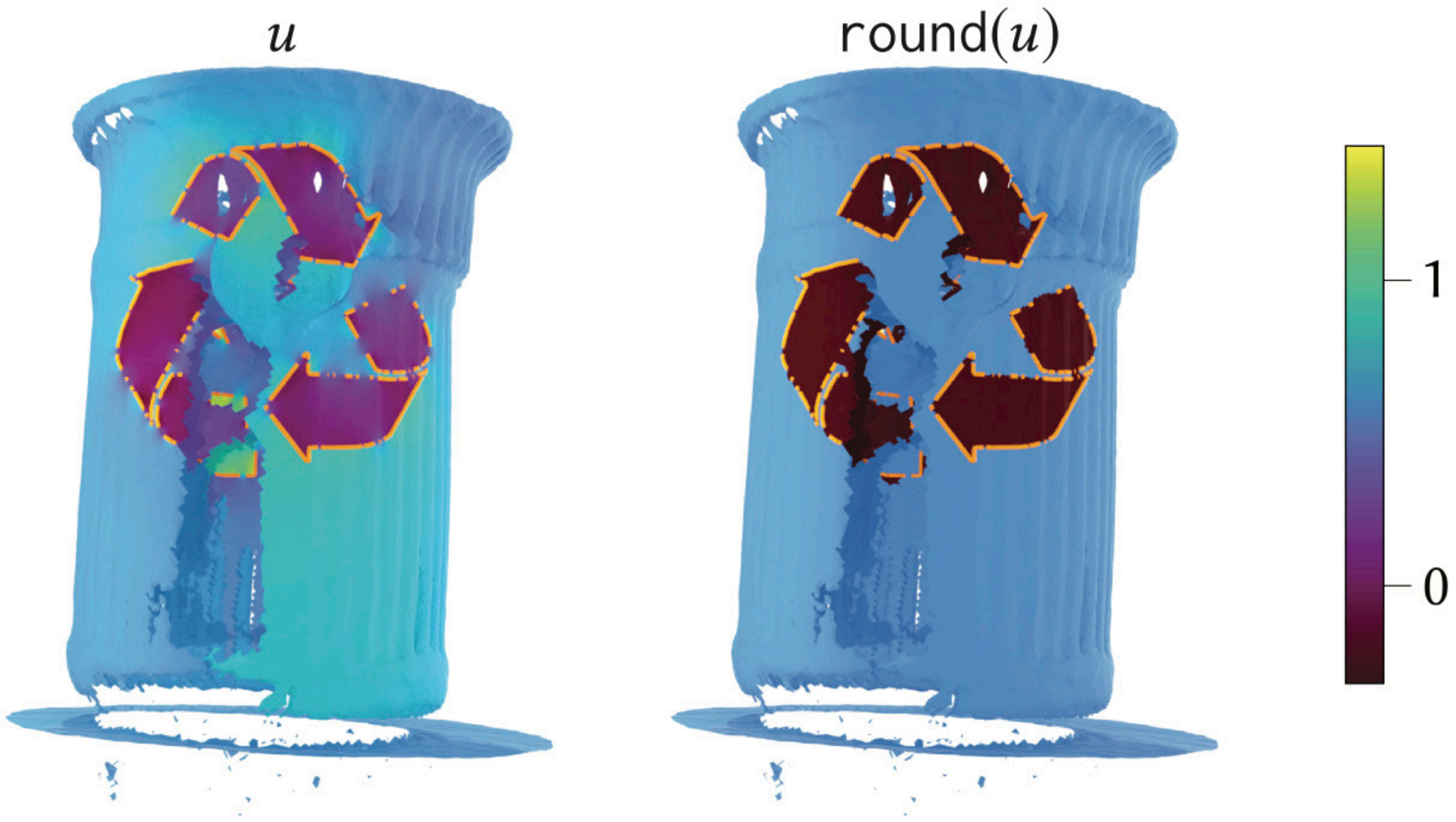


We map from functions back to curves, yielding integer region labels & identification of nonbounding components.

sparse linear program

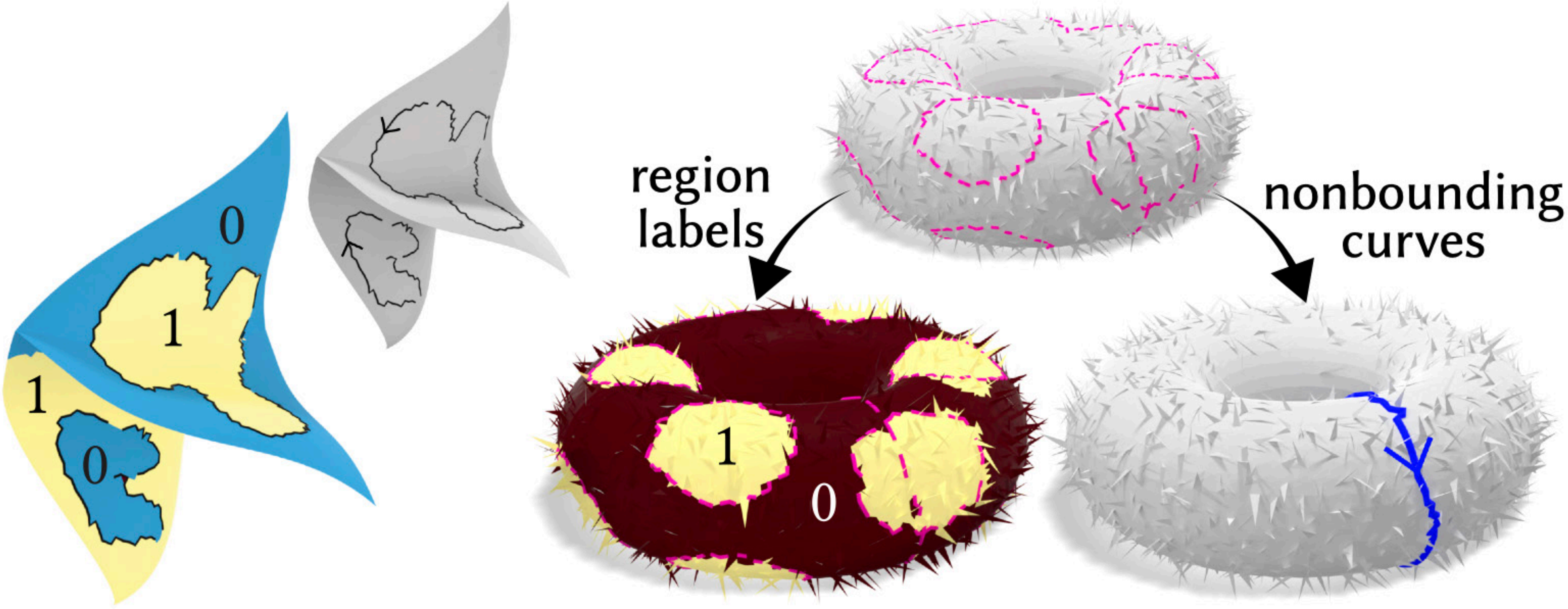
RESULTS

Robustness to defects in both Γ and M



holes, scanner noise

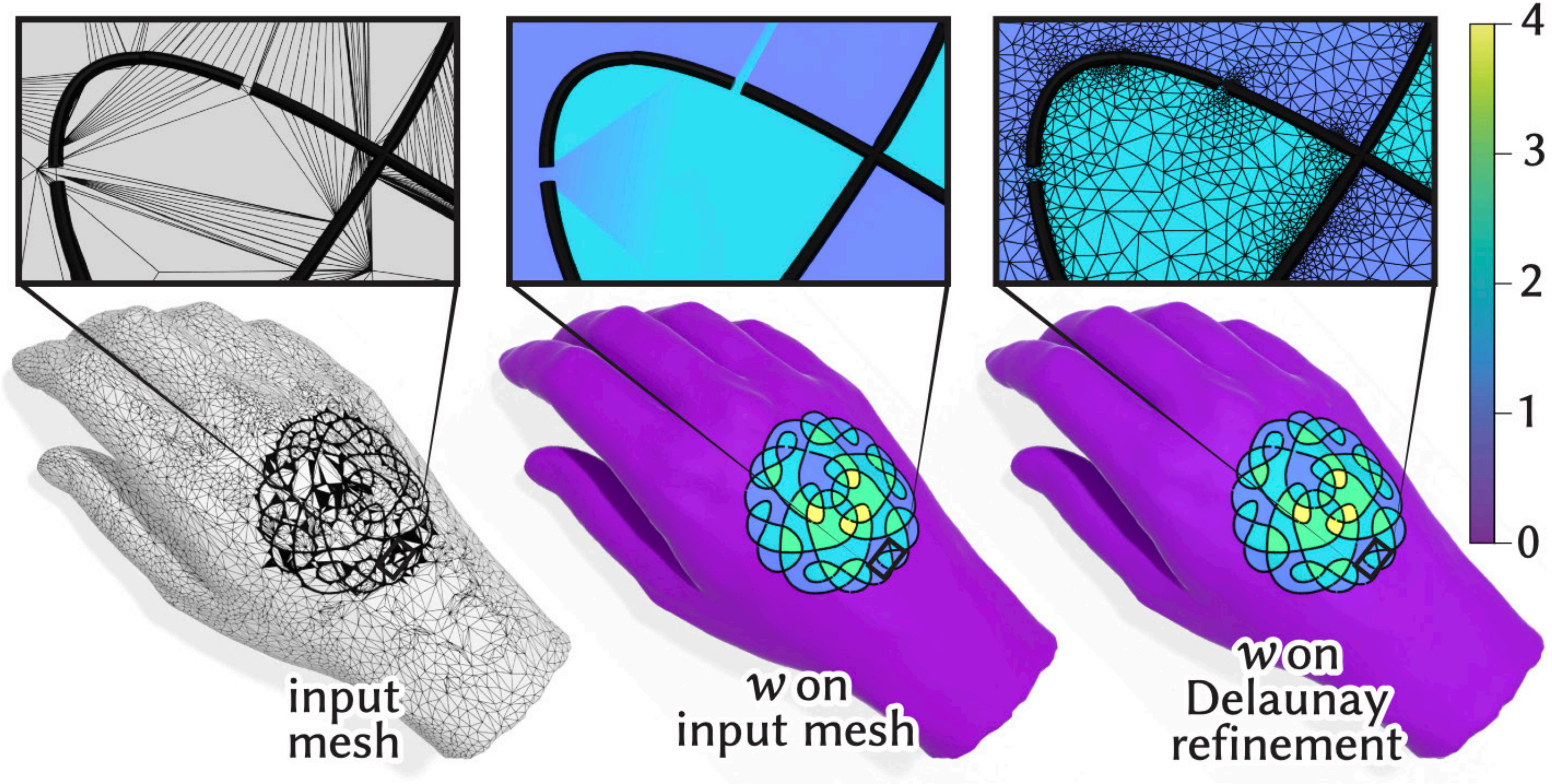
Robustness to defects in both Γ and M



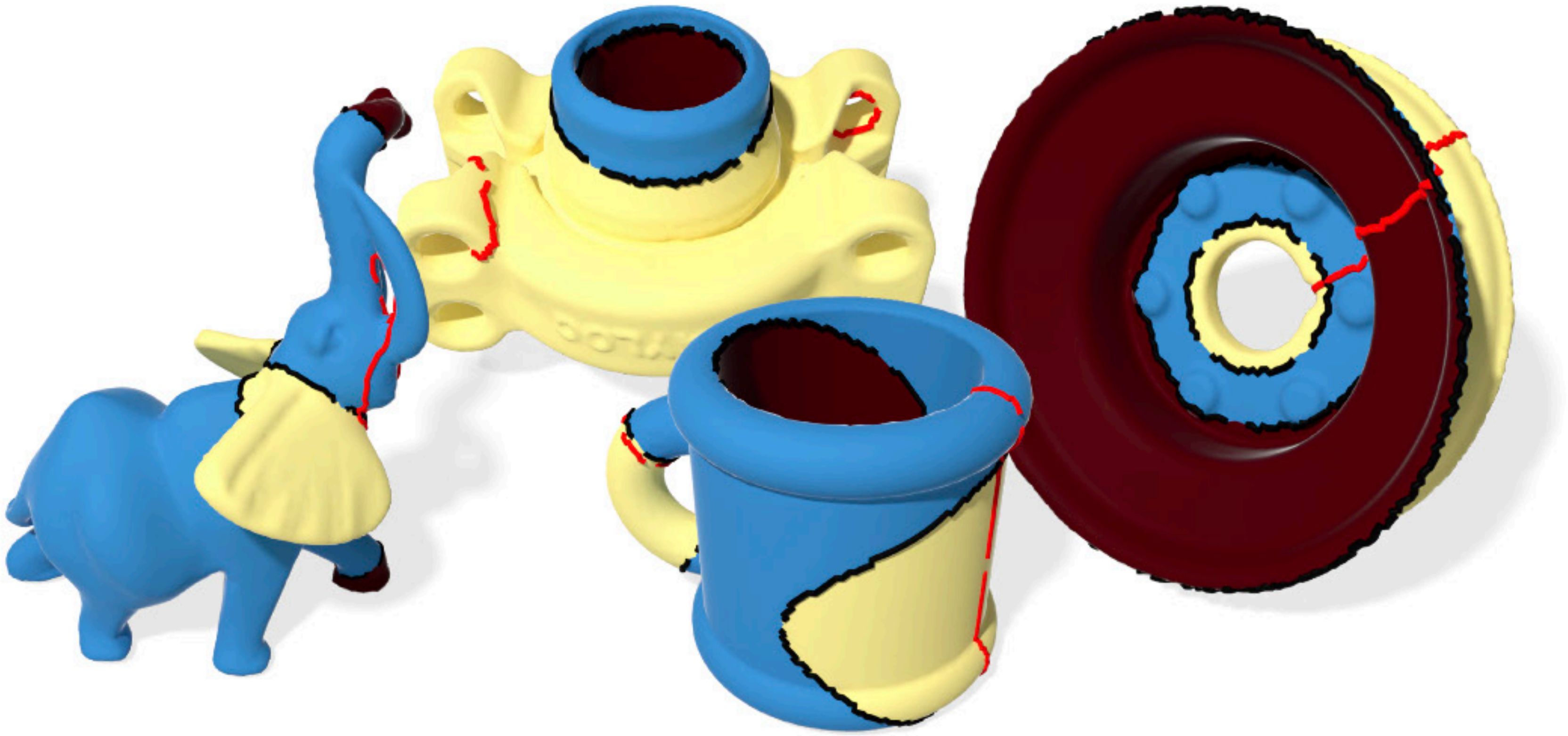
highly non-manifold surfaces

Intrinsic retriangulation

Navigating Intrinsic Triangulations. Sharp, Soliman, Crane (2019)
Integer Coordinates for Intrinsic Geometry Processing. Gillespie, Sharp, Crane (2021)

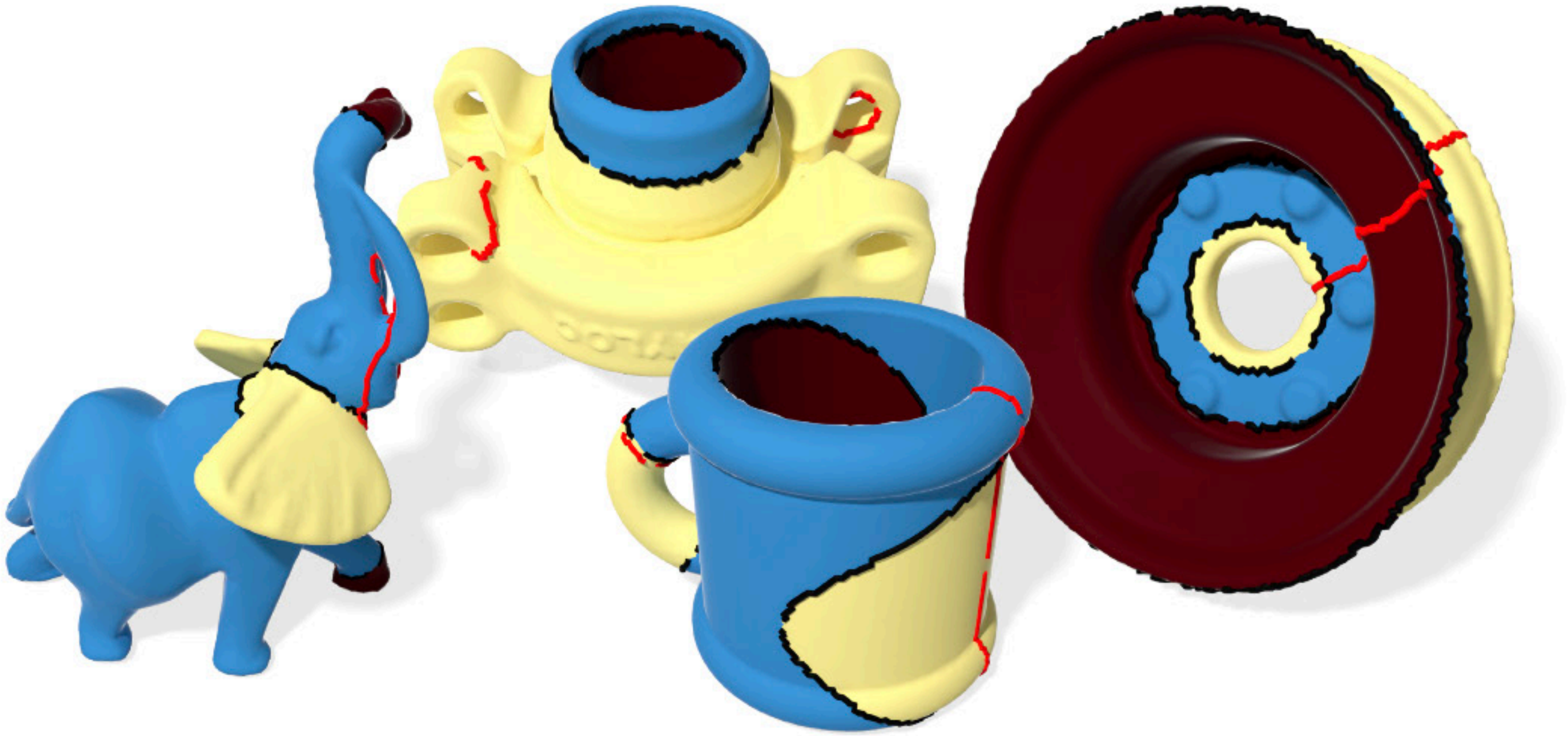


Evaluation



Benchmark setup

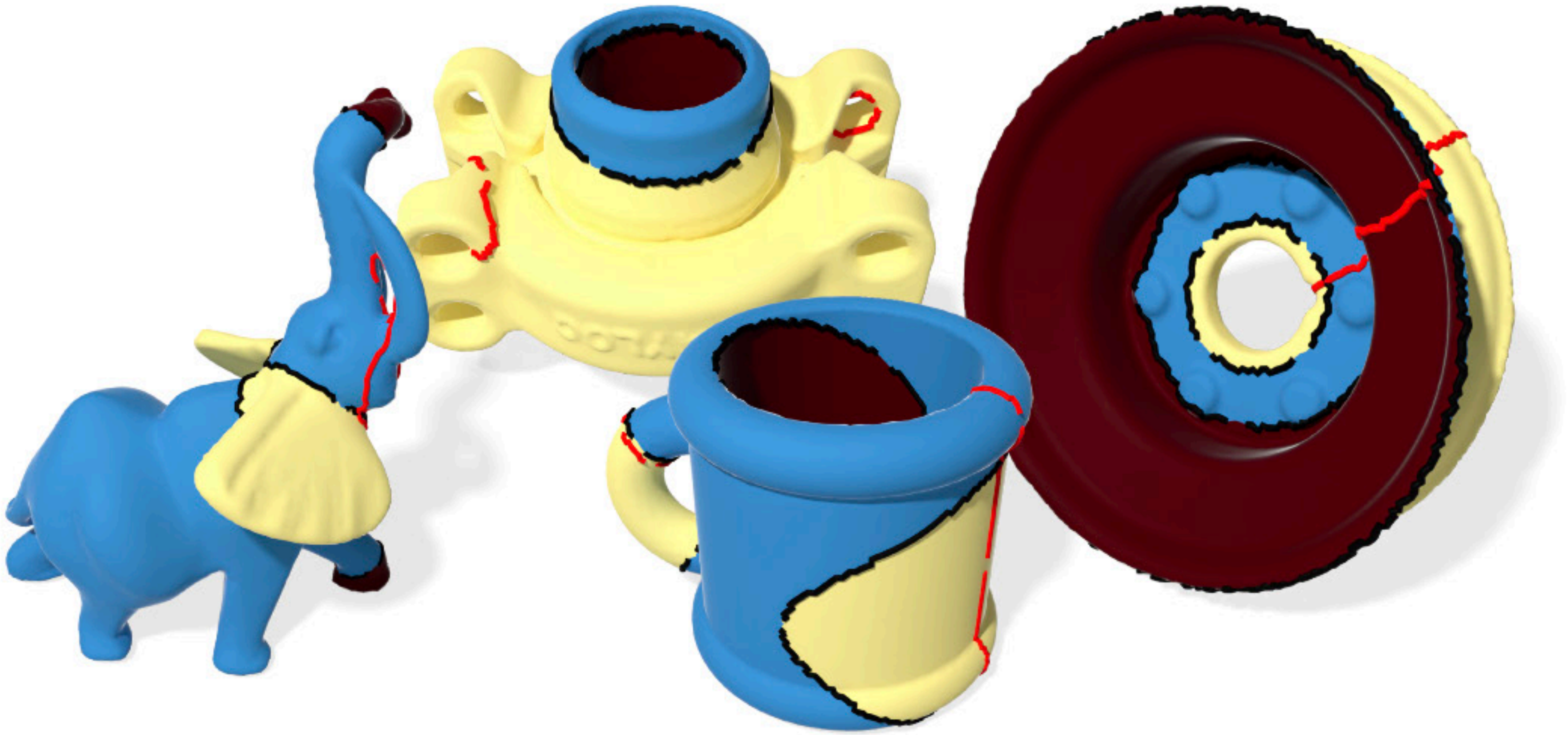
Evaluation



Benchmark setup

(1) Generate random ground-truth regions, take boundaries *[sub-levelsets of low-frequency Laplacian eigenfunctions]*

Evaluation

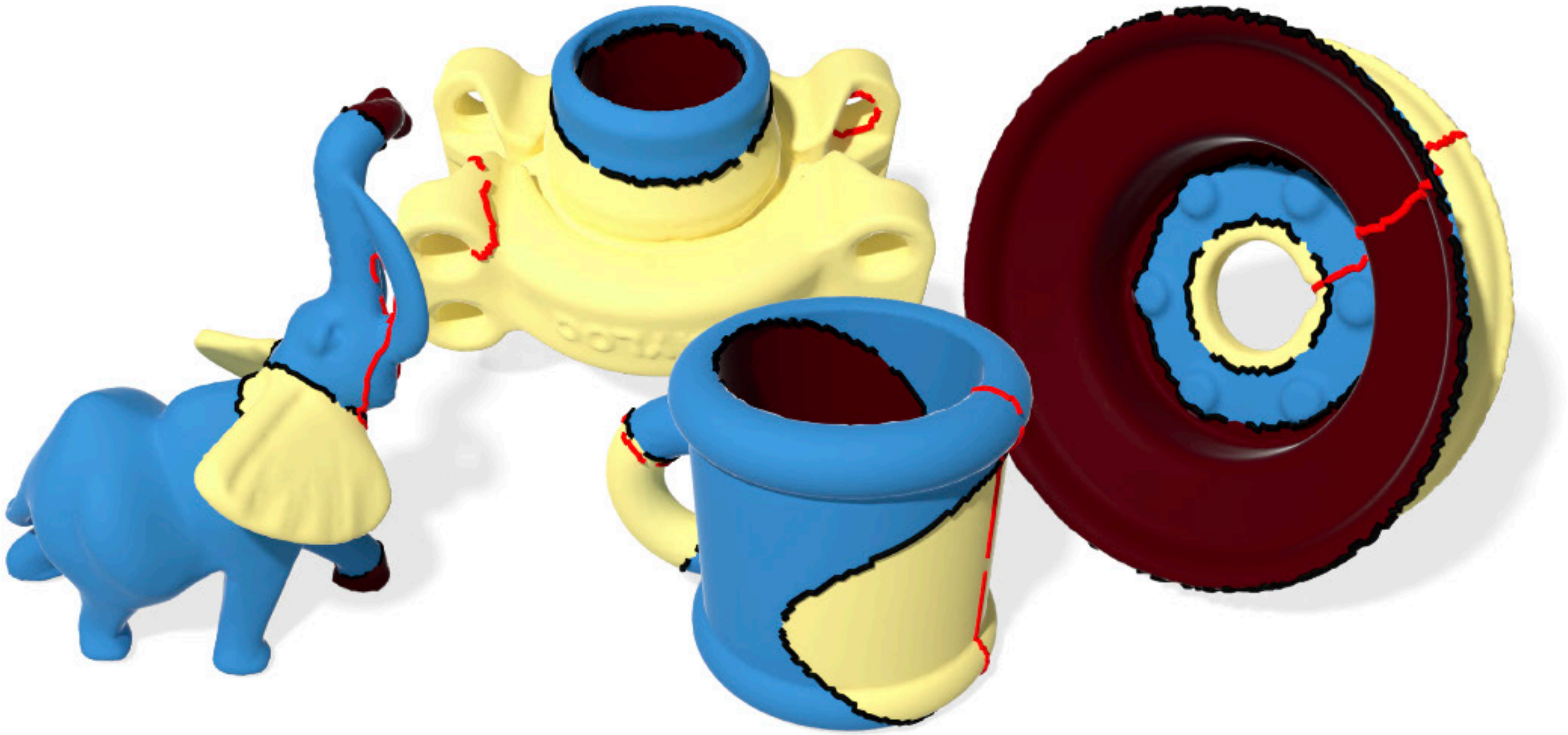


Benchmark setup

(1) Generate random ground-truth regions, take boundaries *[sub-levelsets of low-frequency Laplacian eigenfunctions]*

(2) Add nonbounding loops *[compute homology basis, select random subset]*

Evaluation



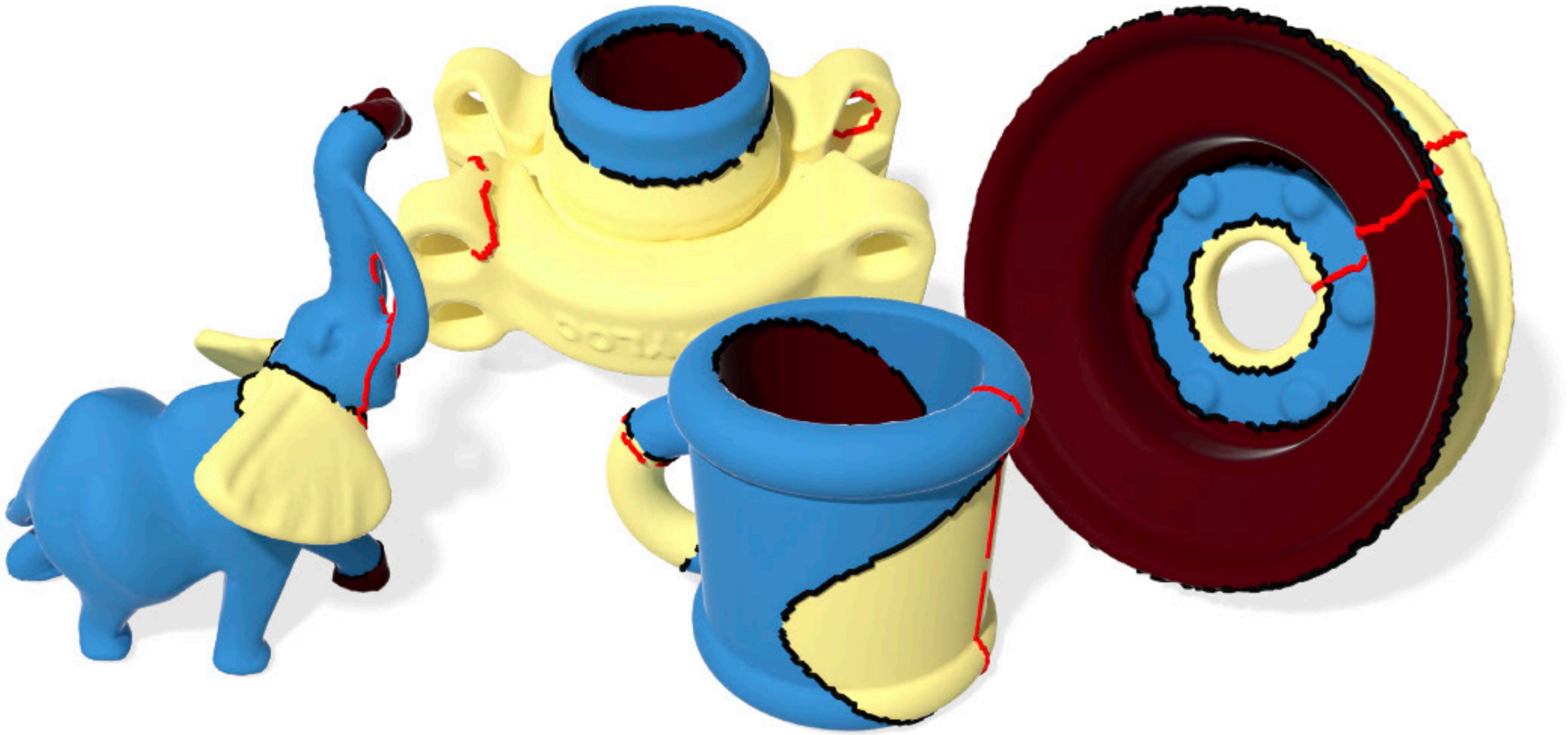
Benchmark setup

(1) Generate random ground-truth regions, take boundaries *[sub-levelsets of low-frequency Laplacian eigenfunctions]*

(2) Add nonbounding loops *[compute homology basis, select random subset]*

(3) Break up curves *[use random gap & dash size]*

Evaluation



Benchmark setup

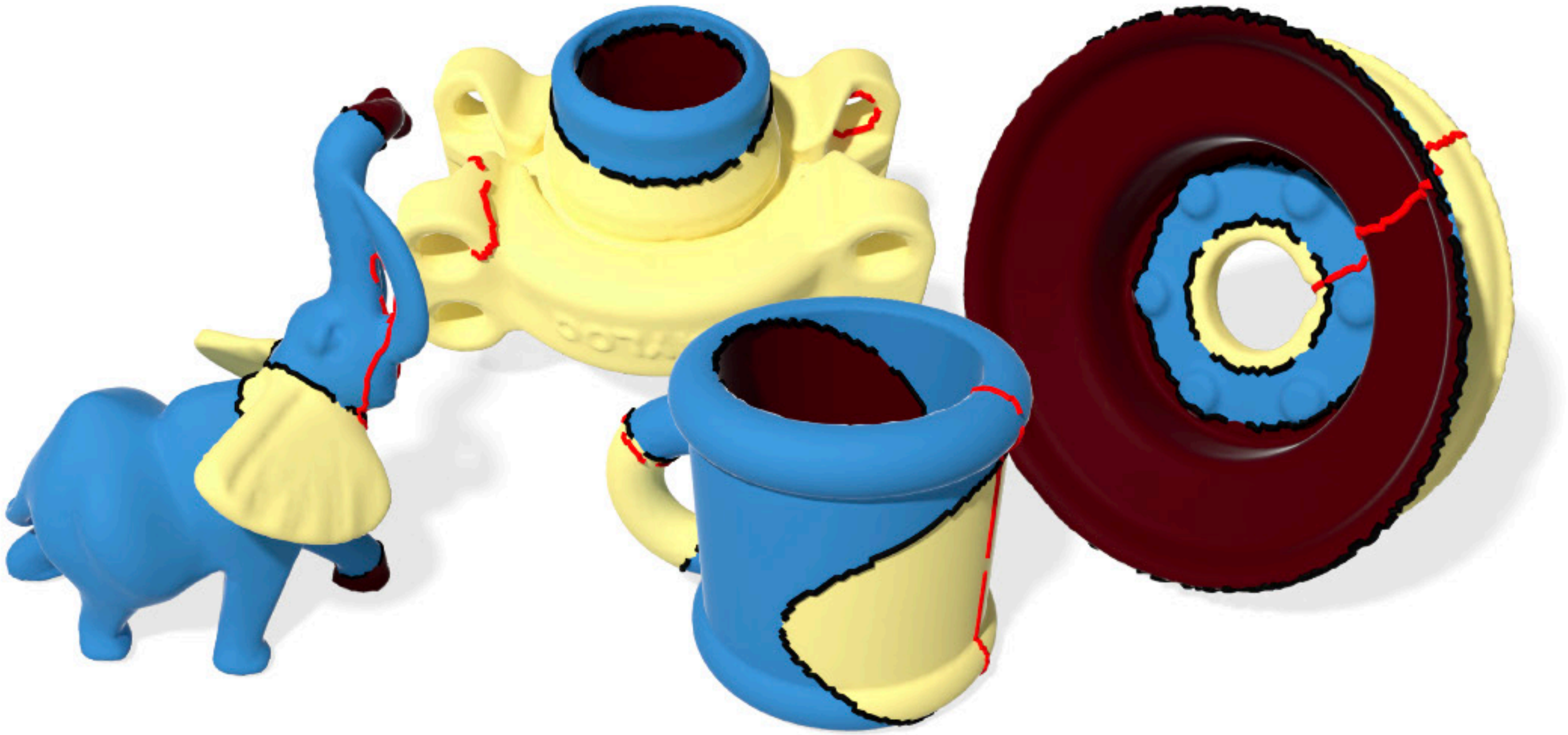
(1) Generate random ground-truth regions, take boundaries *[sub-levelsets of low-frequency Laplacian eigenfunctions]*

(2) Add nonbounding loops *[compute homology basis, select random subset]*

(3) Break up curves *[use random gap & dash size]*

(4) Run SWN; compute % of surface area correctly classified *[shift w to match ground-truth value in an arbitrary face]*

Evaluation



Benchmark setup

(1) Generate random ground-truth regions, take boundaries *[sub-levelsets of low-frequency Laplacian eigenfunctions]*

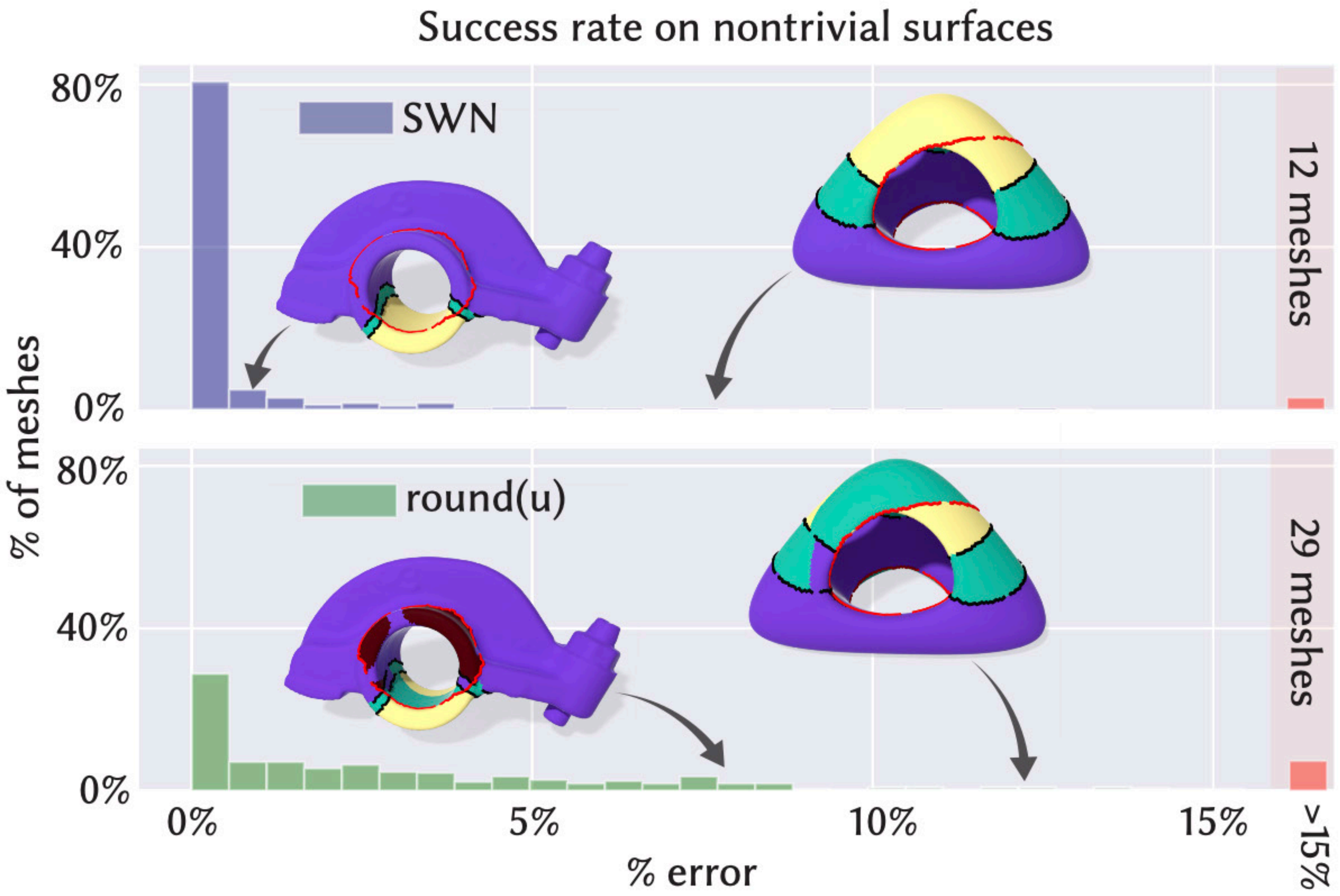
(2) Add nonbounding loops *[compute homology basis, select random subset]*

(3) Break up curves *[use random gap & dash size]*

(4) Run SWN; compute % of surface area correctly classified *[shift w to match ground-truth value in an arbitrary face]*

934 total test cases, 451 multiply-connected

Evaluation



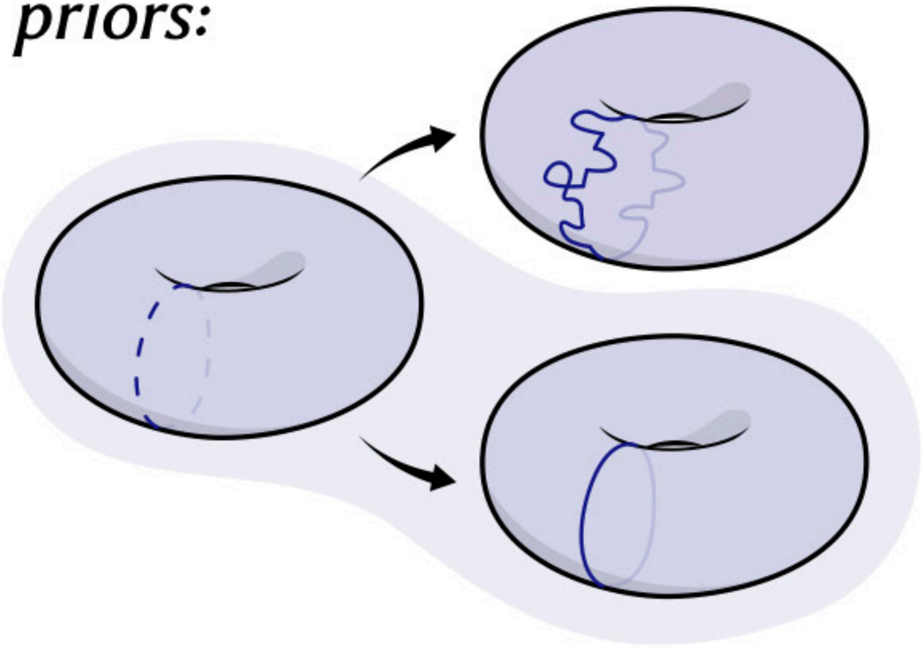
Ours

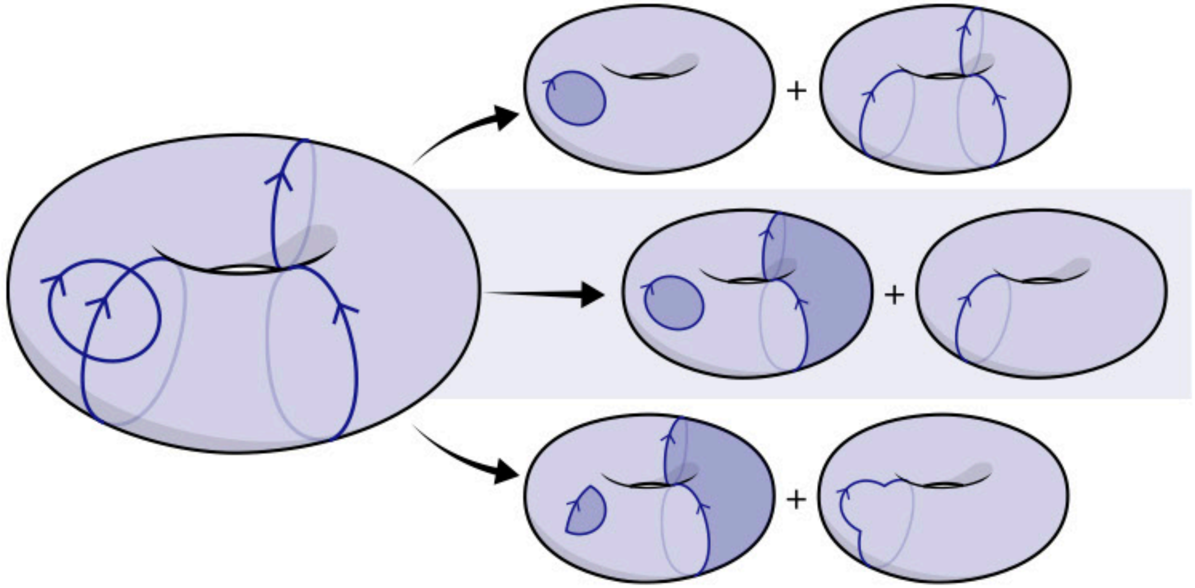
PSR/GWN

Much less accurate without homology processing

“Failure” cases

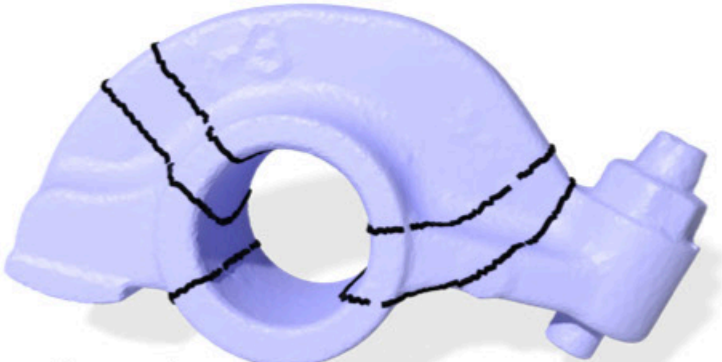
Recall shortest-length priors:



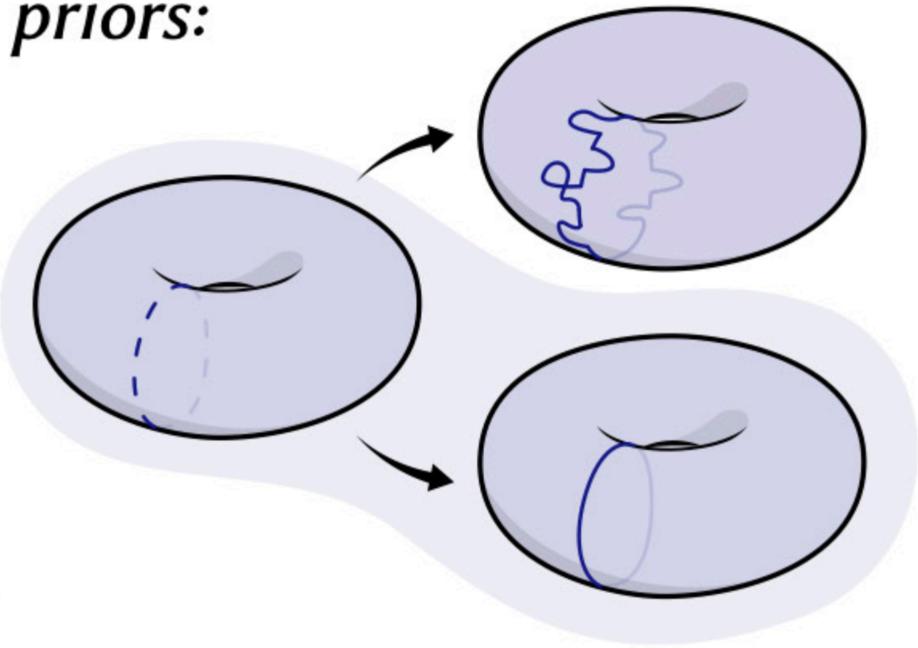


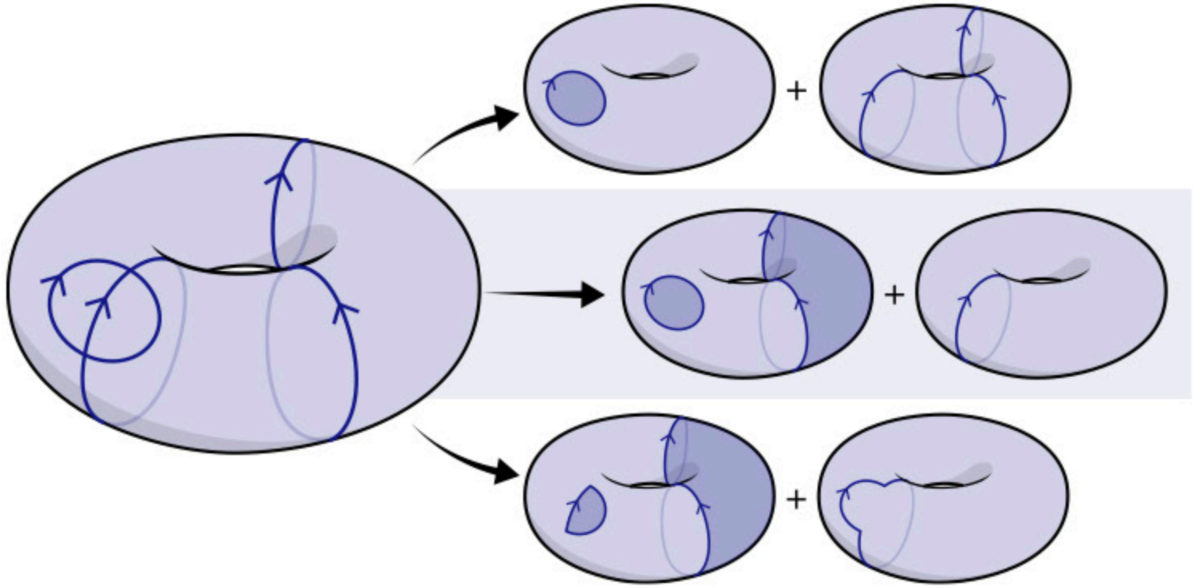
“Failure” cases

Recall shortest-length priors:



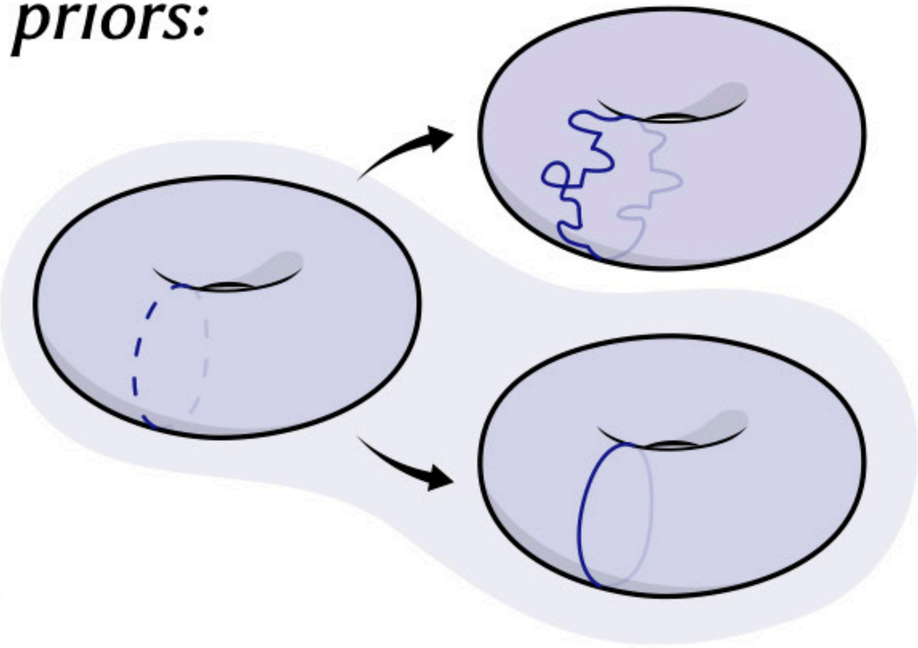
input

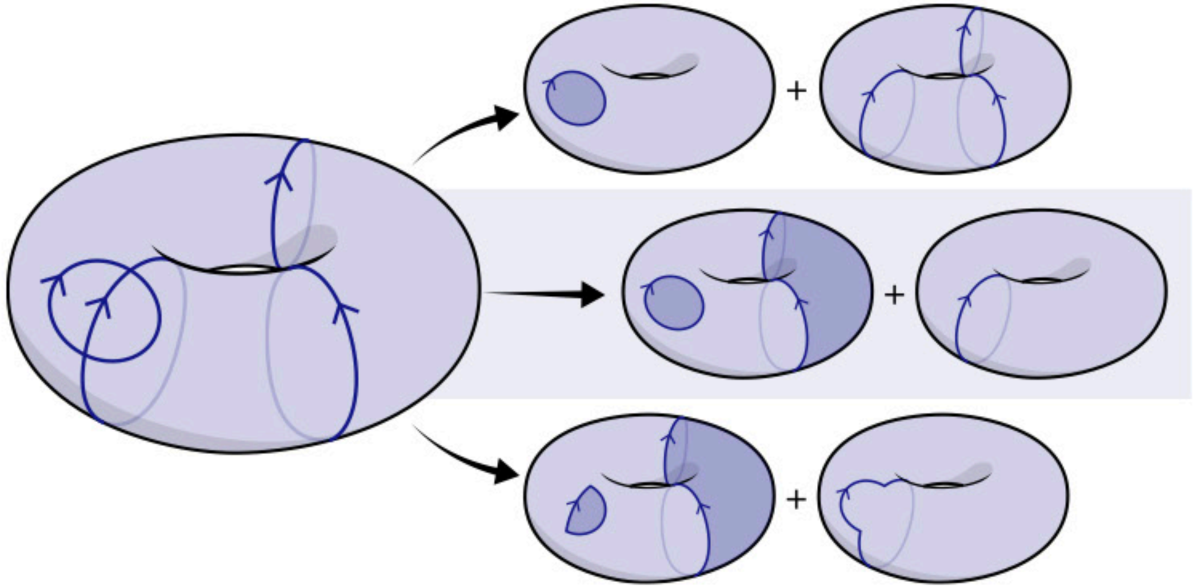


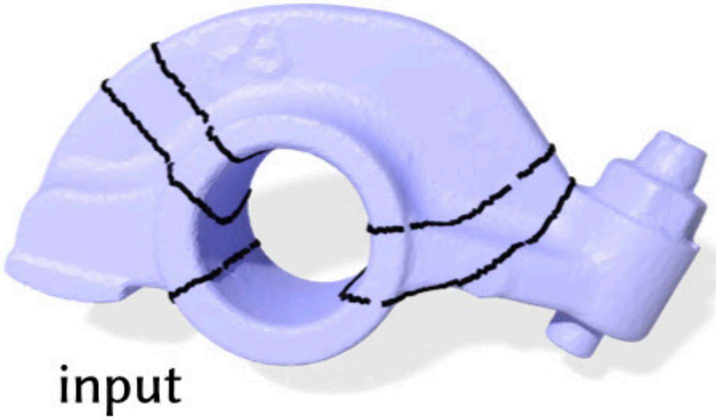


“Failure” cases

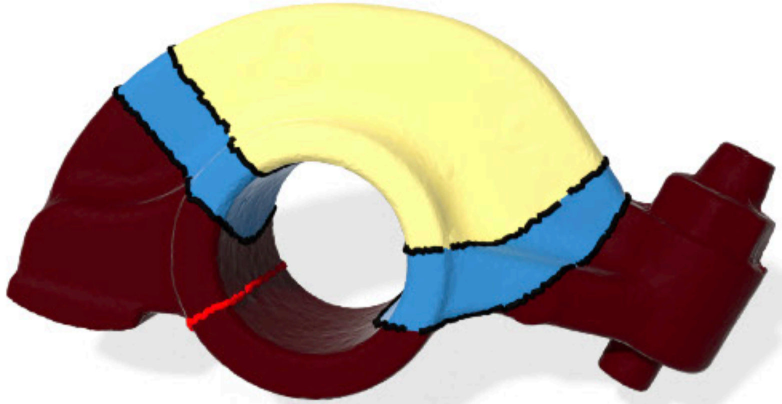
Recall shortest-length priors:



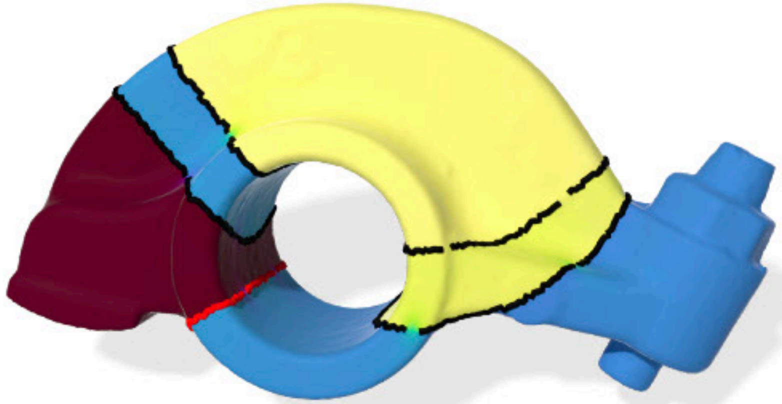




input



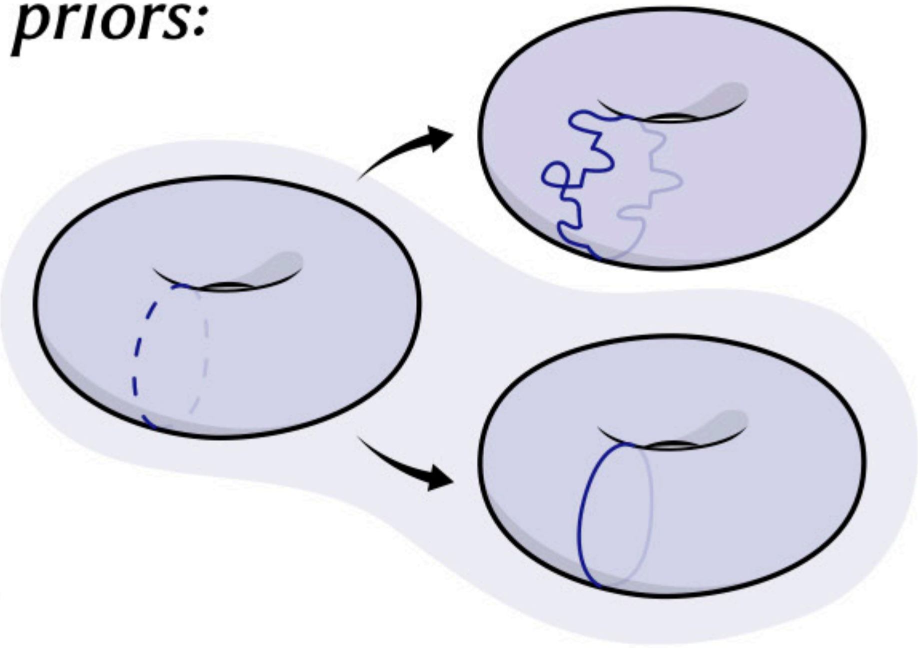
ground truth regions

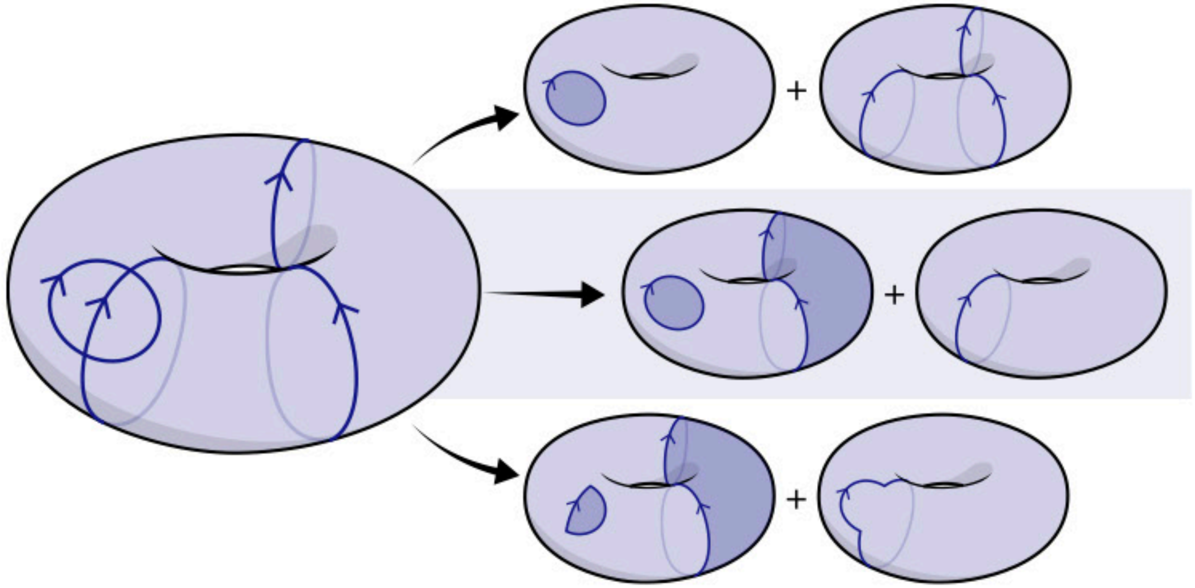


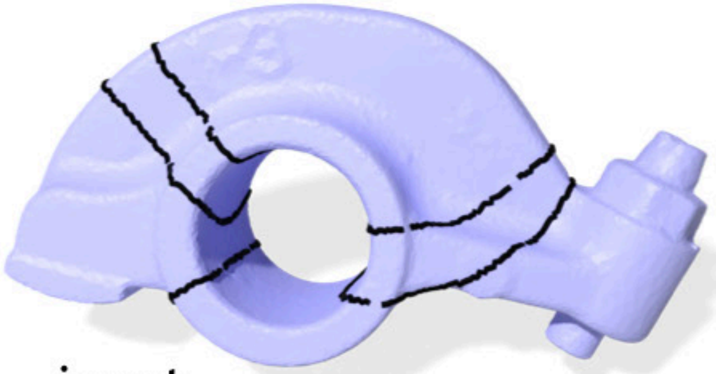
SWN

“Failure” cases

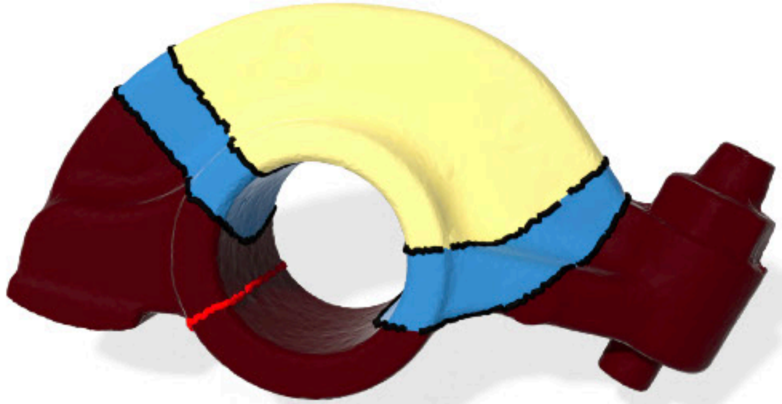
Recall shortest-length priors:



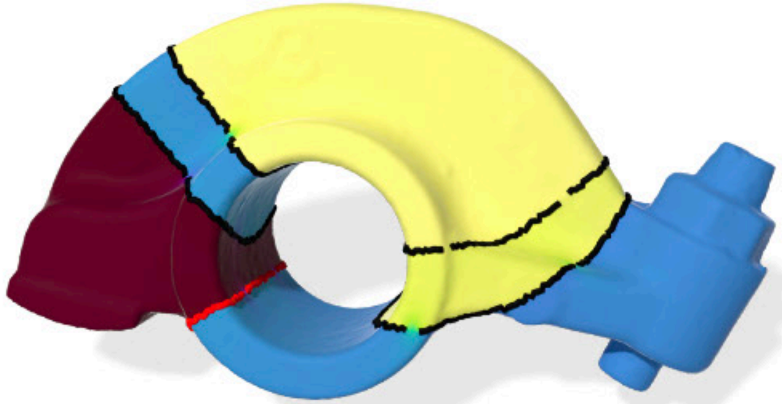




input



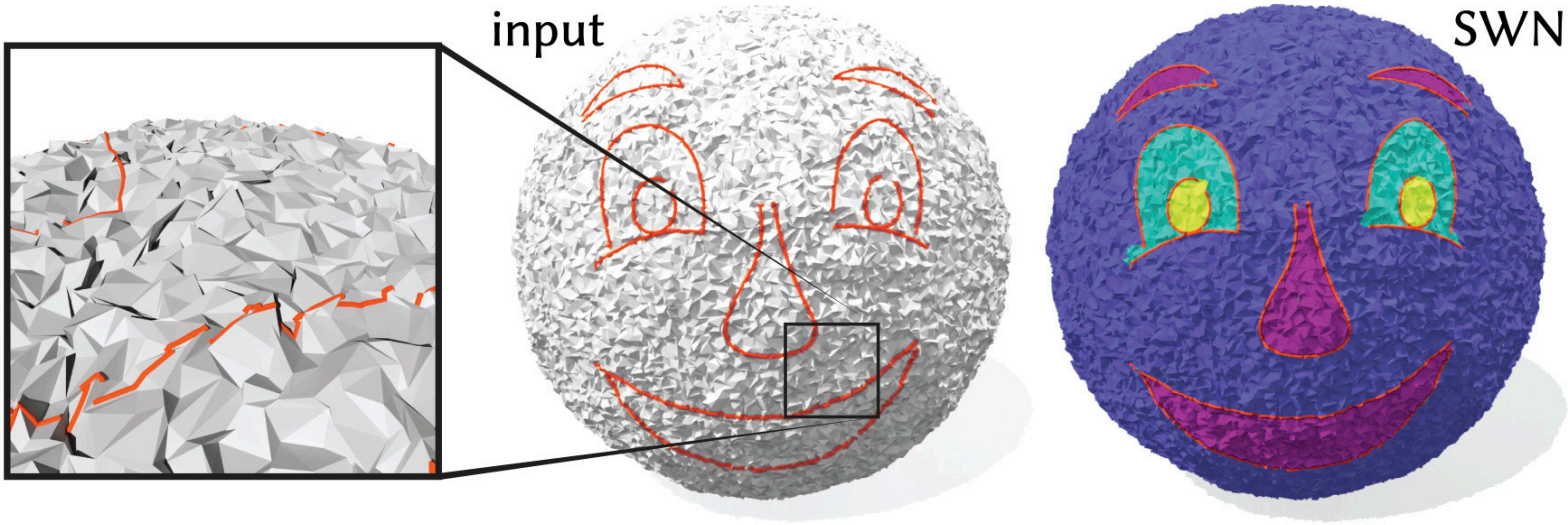
ground truth regions



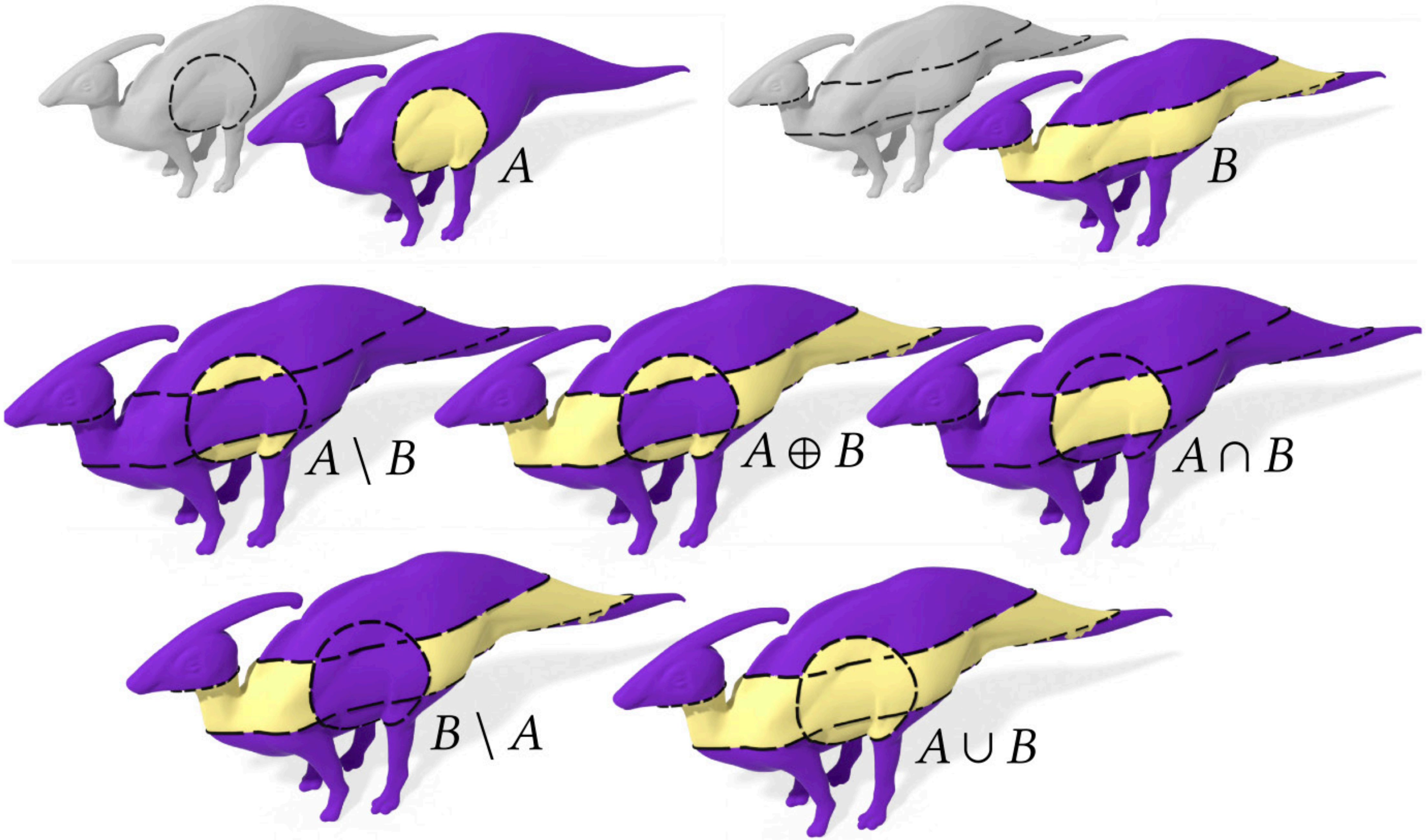
SWN

SWN still gives a valid curve decomposition!
(no single “right” answer.)

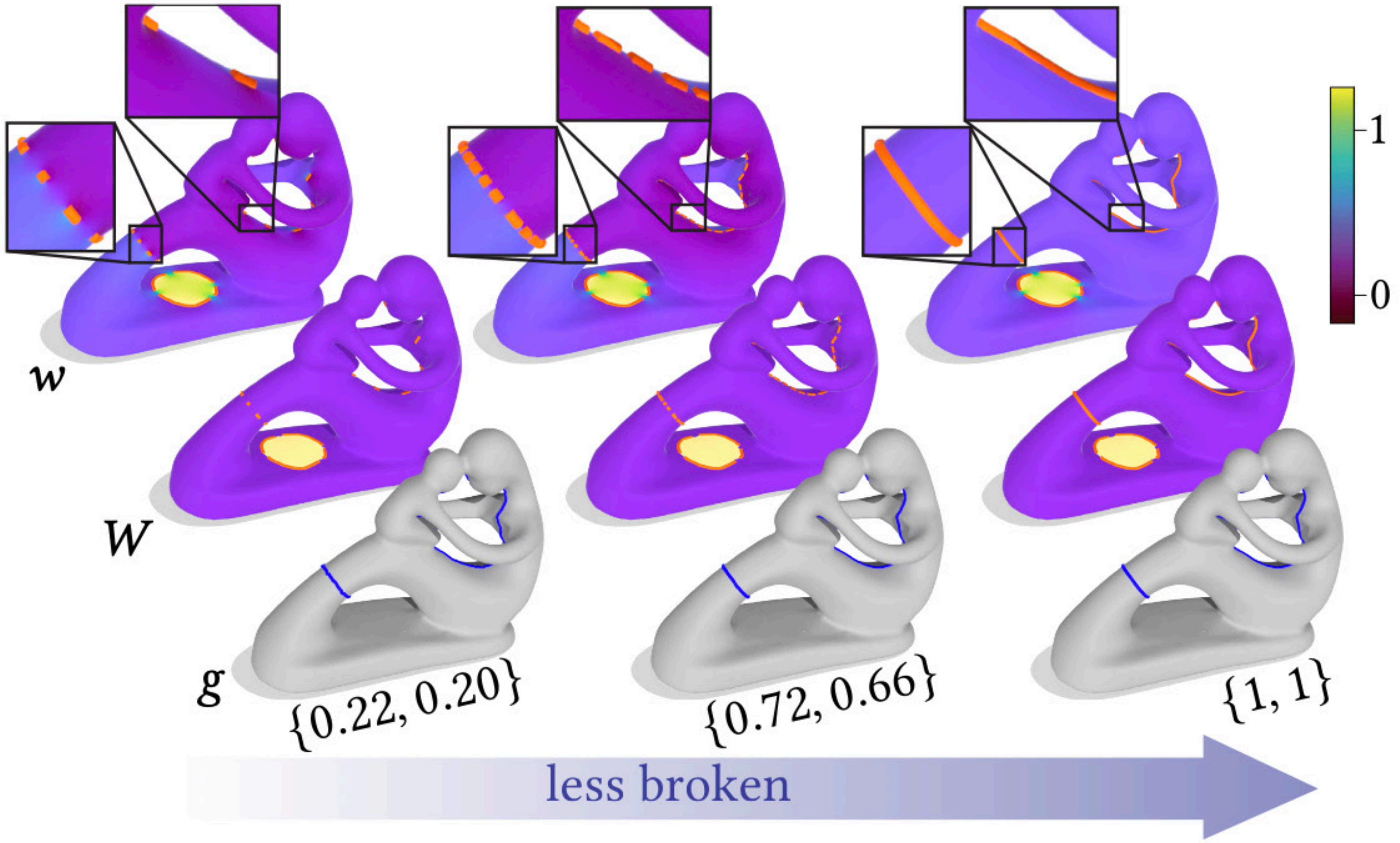
Surface sketching



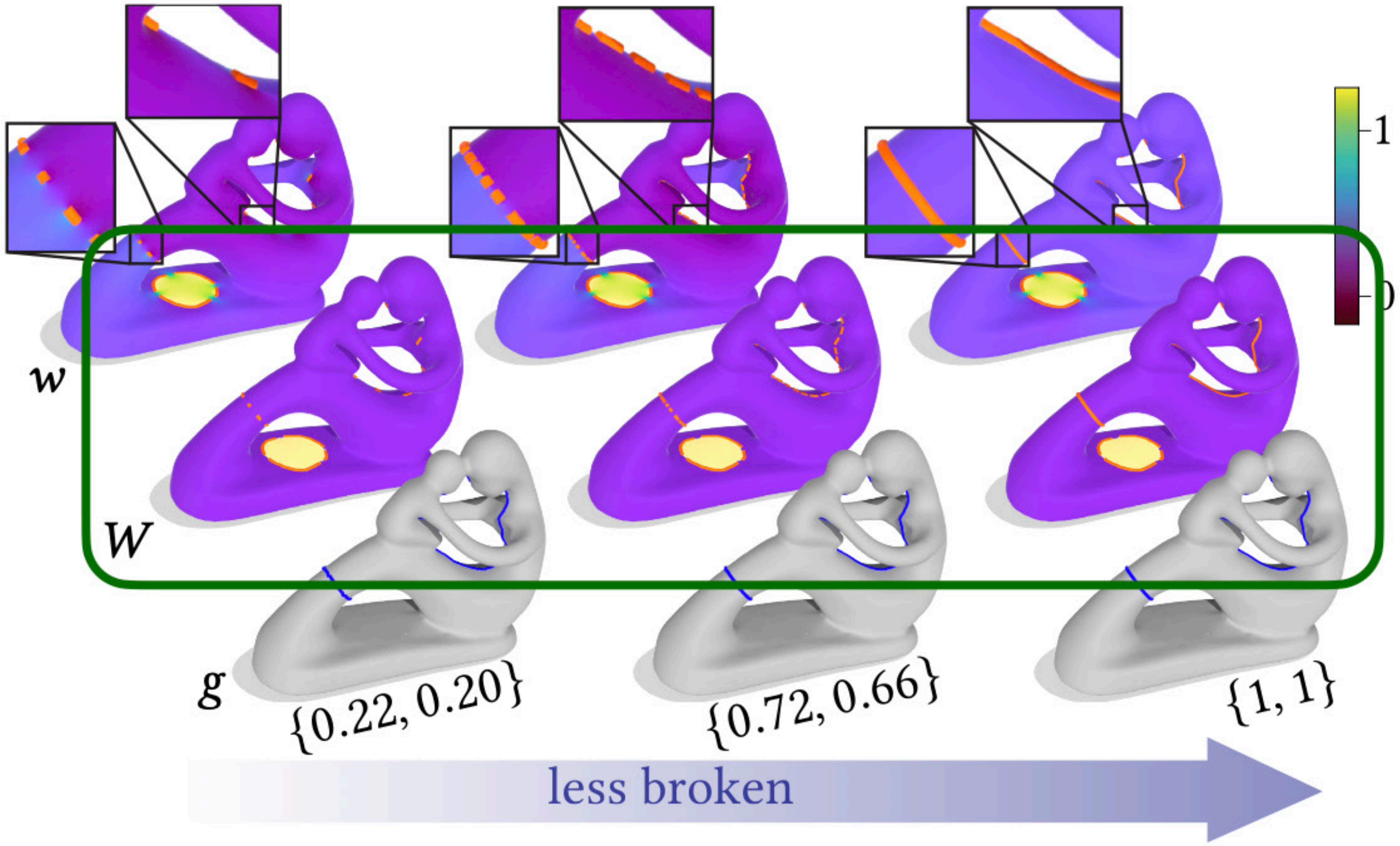
Booleans



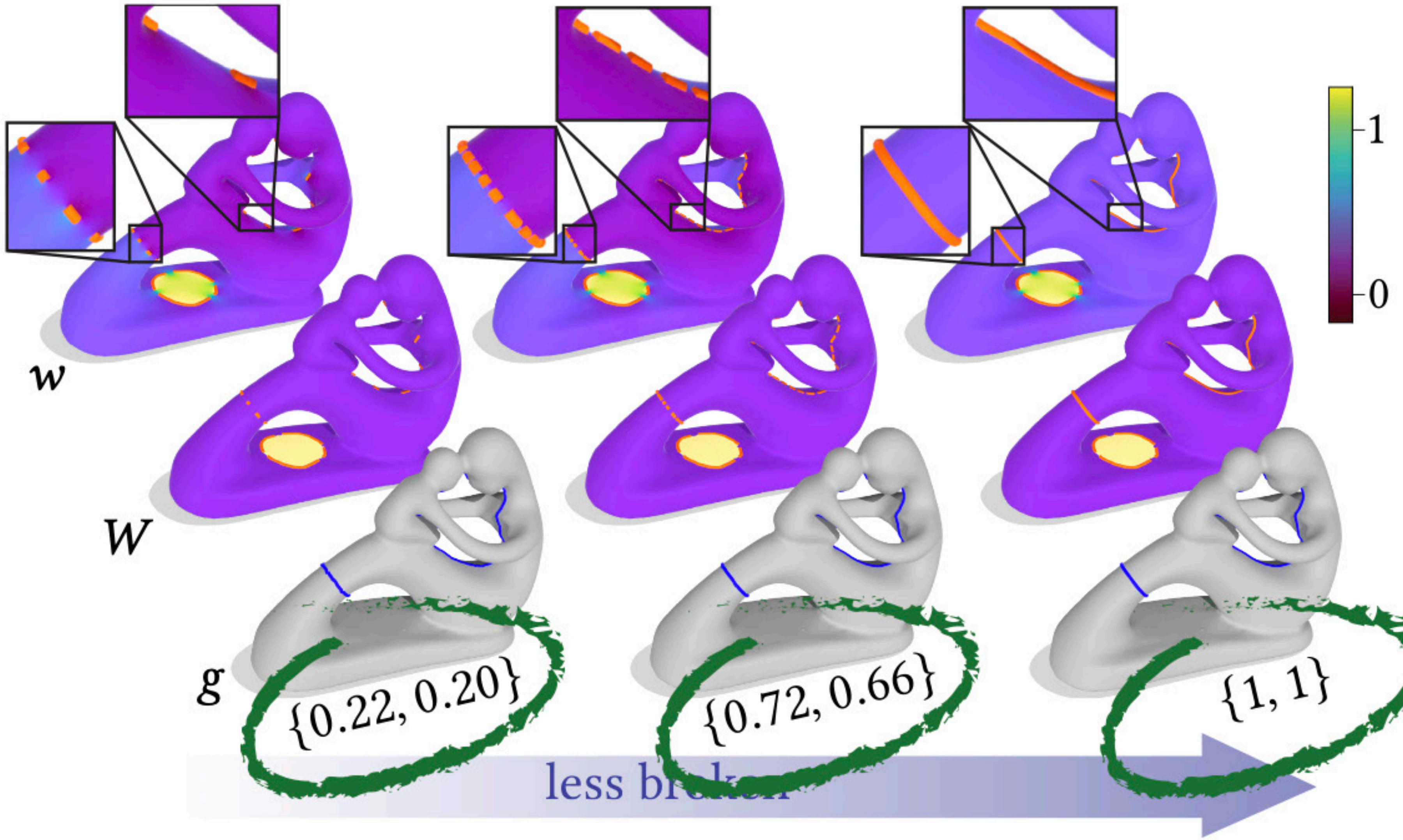
Curve decomposition



Curve decomposition

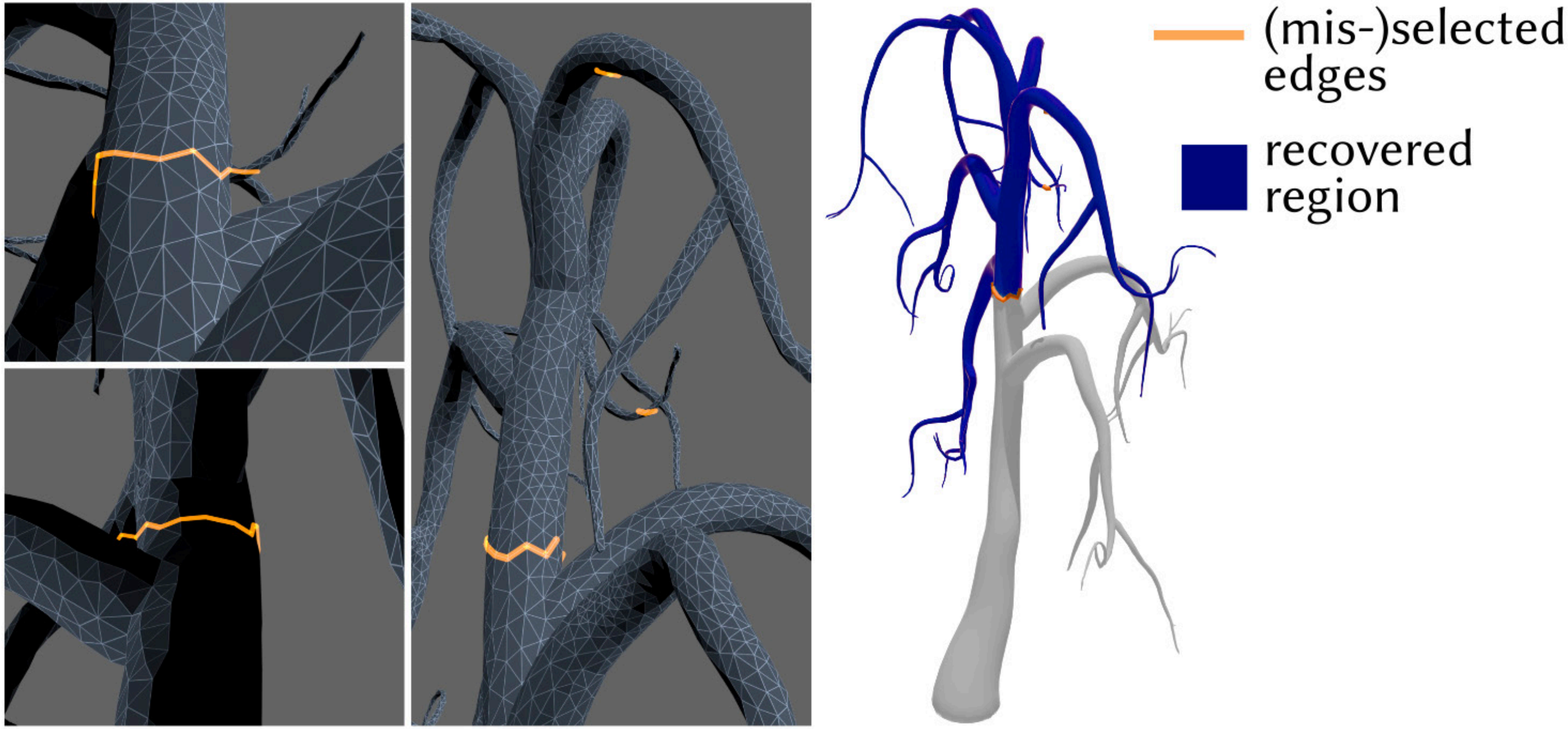


Curve decomposition



g values give SWN's confidence in nonbounding loops

Region selection — robustness

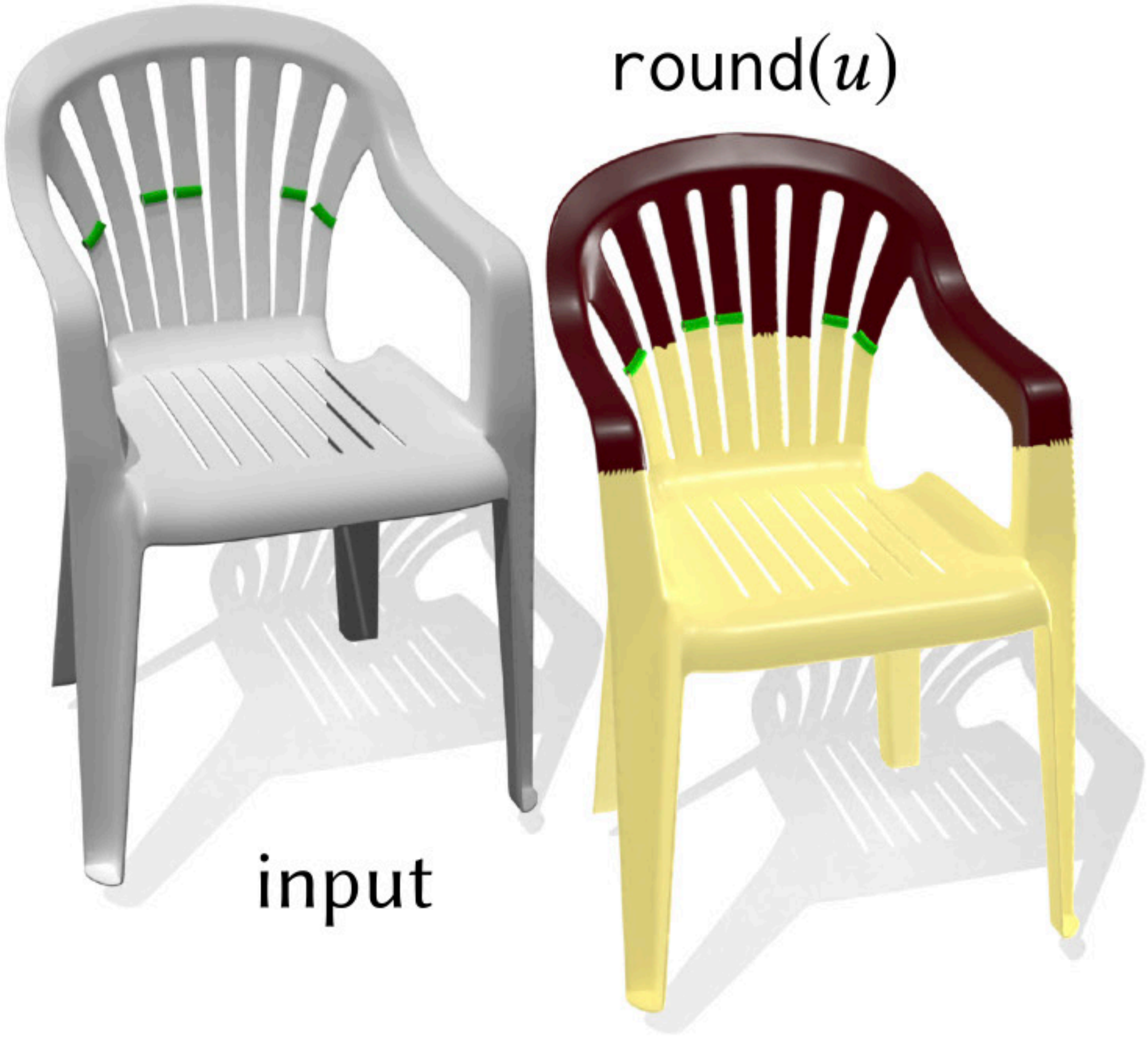


Region selection — automatic completion

input

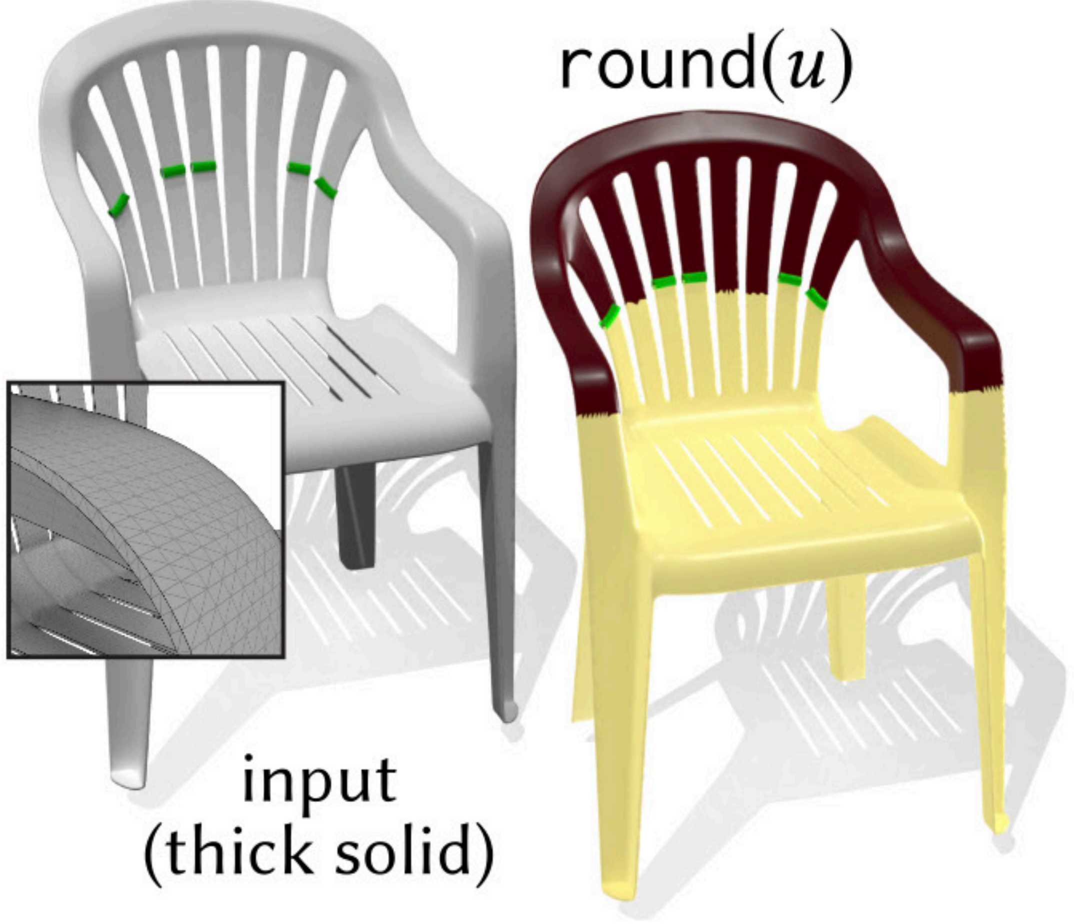
Region selection — automatic completion

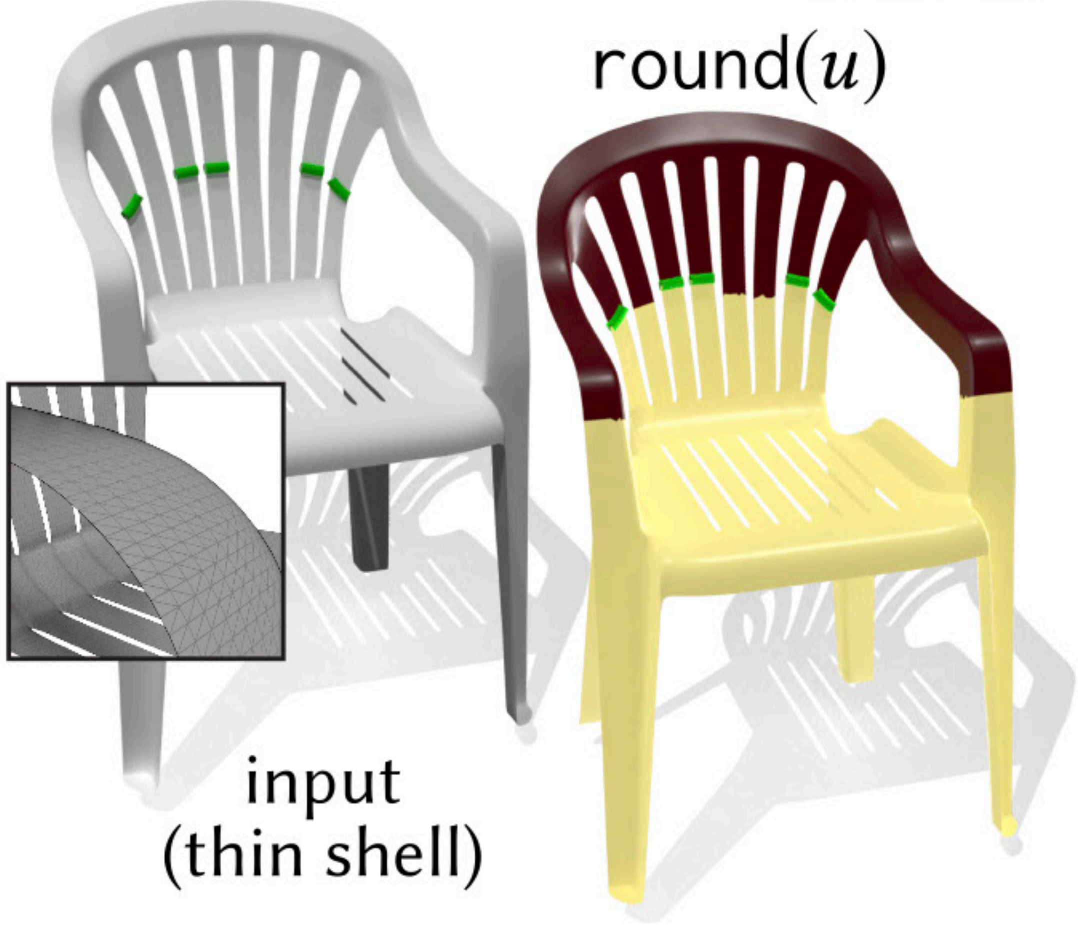
using "ghost loops" to *aid* segmentation



Region selection — automatic completion

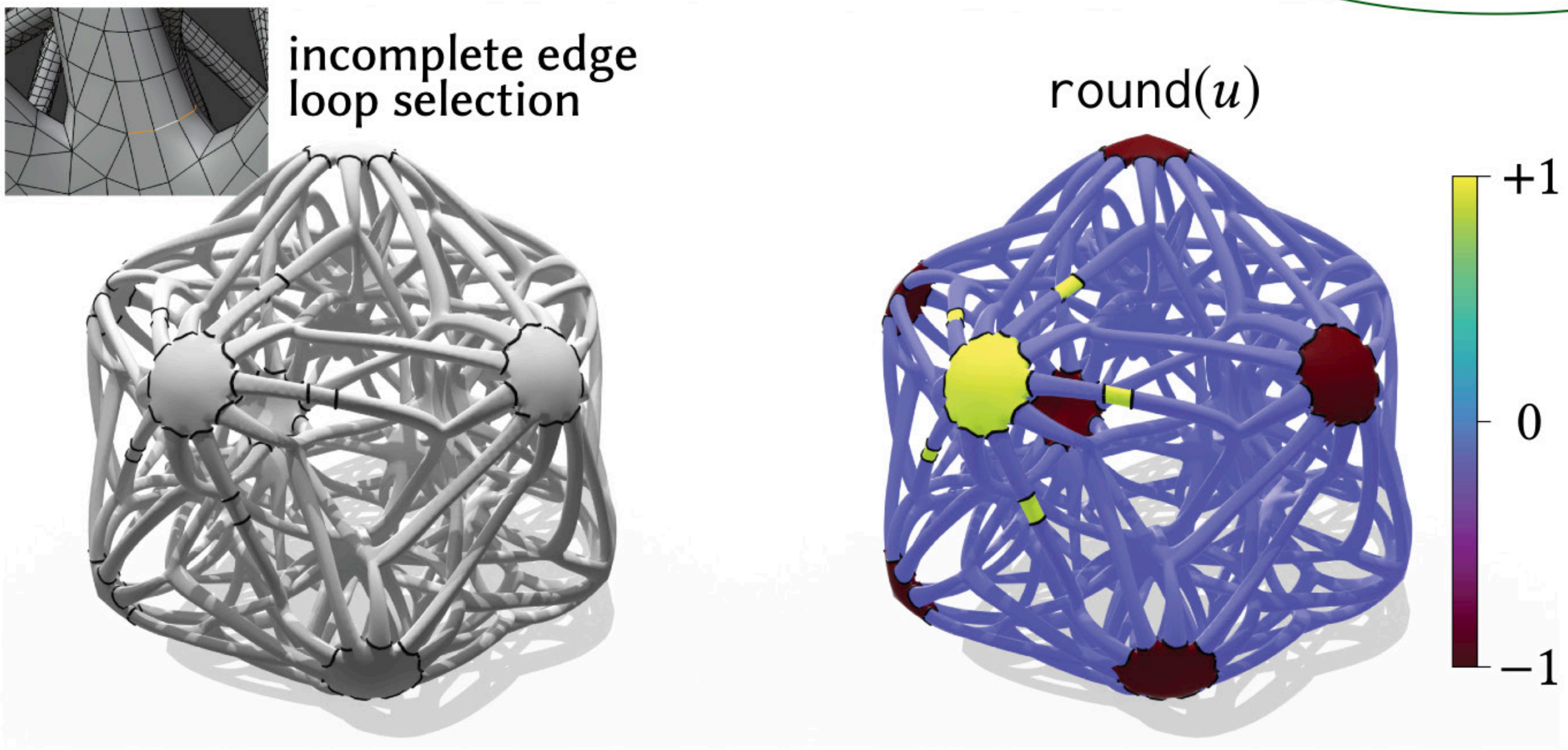
using "ghost loops"
to *aid* segmentation





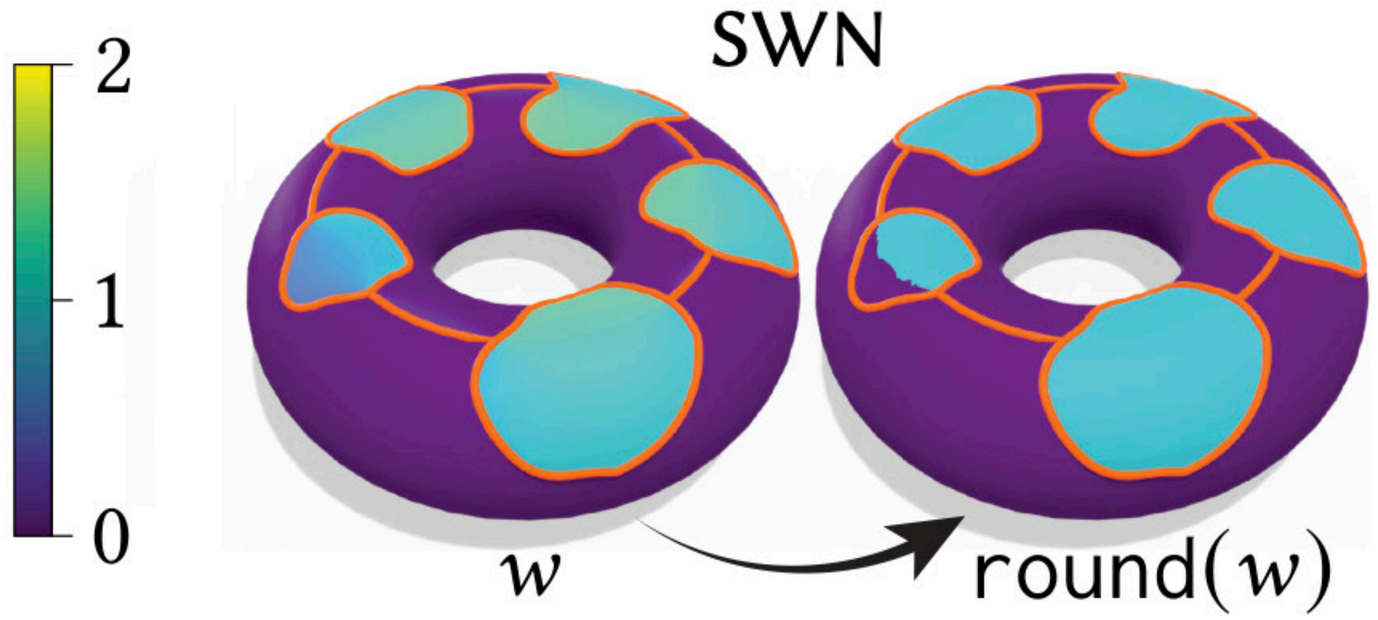
Region selection — automatic completion

using "ghost loops" to *aid* segmentation



LIMITATIONS & FUTURE WORK

Contouring is sometimes counterintuitive



Contouring is sometimes counterintuitive

Contouring is sometimes counterintuitive

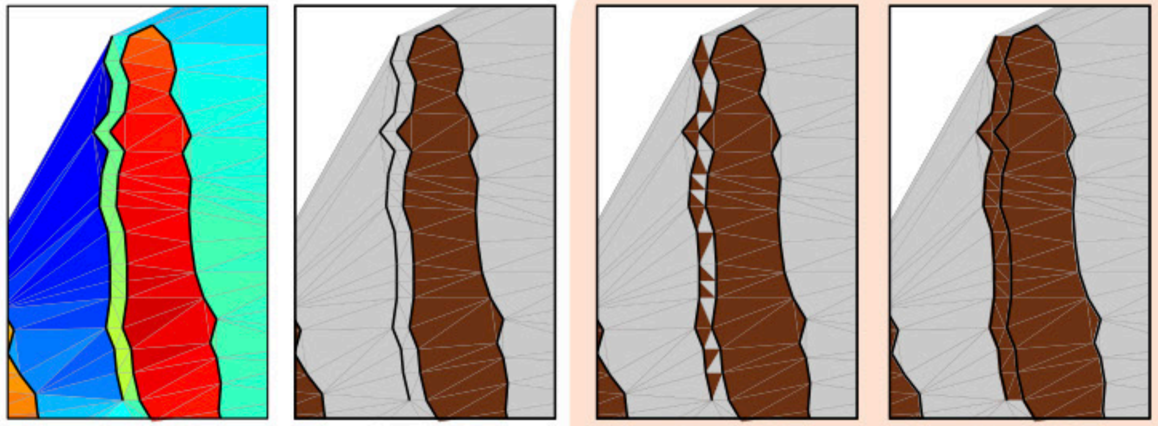
Euclidean winding number also struggles

Jacobson et al. suggest a graphcut algorithm for contouring:

Robust Inside-Outside Segmentation using Generalized Winding Numbers
Jacobson, Kavan, Sorkine-Hornung (2013)

Euclidean winding number also struggles

Jacobson et al. suggest a graphcut algorithm for contouring:



GWN

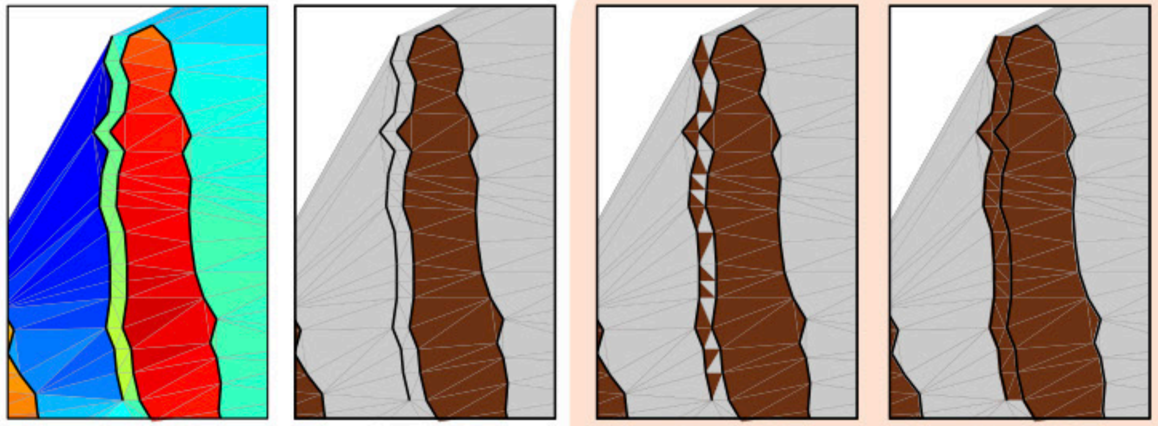
naive contouring

graphcut

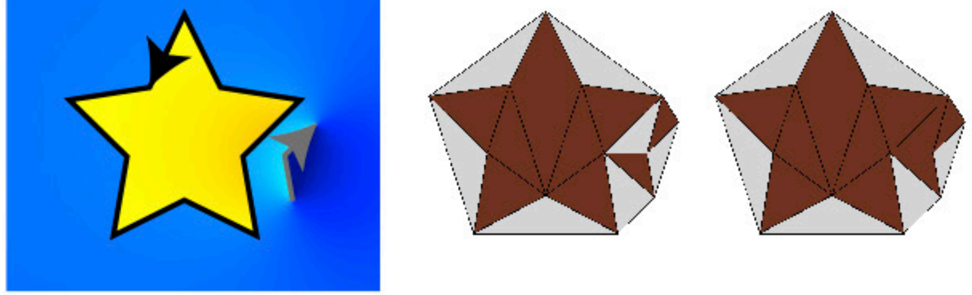
Robust Inside-Outside Segmentation using Generalized Winding Numbers
Jacobson, Kavan, Sorkine-Hornung (2013)

Euclidean winding number also struggles

Jacobson et al. suggest a graphcut algorithm for contouring:



GWN naive contouring graphcut



GWN contoured

Still not perfect!

Robust Inside-Outside Segmentation using Generalized Winding Numbers
Jacobson, Kavan, Sorkine-Hornung (2013)

Unexpected nonbounding loops

Recall objective function: $\min_{v:\ M \to \mathbb{R}} \int |\text{the jumps not across } \Gamma| + \varepsilon \int |\text{the jumps across } \Gamma|$

Unexpected nonbounding loops

Recall objective function: $\min_{v:\, M \to \mathbb{R}} \int |\text{the jumps not across } \Gamma| + \boxed{\varepsilon \int |\text{the jumps across } \Gamma|}$

encourage jumps across Γ

Unexpected nonbounding loops

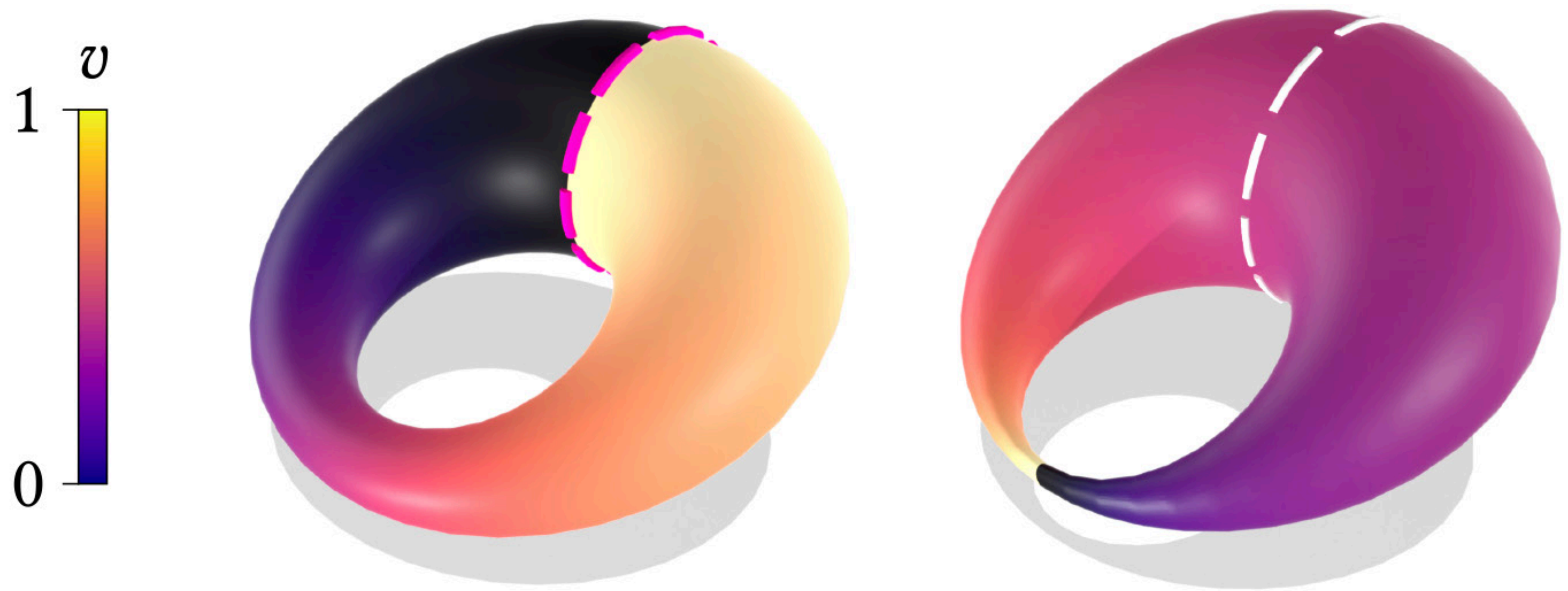
Recall objective function: $\min_{v:\, M \to \mathbb{R}} \int |\text{the jumps not across } \Gamma| + \varepsilon \int |\text{the jumps across } \Gamma|$

encourage jumps across Γ

Unexpected nonbounding loops

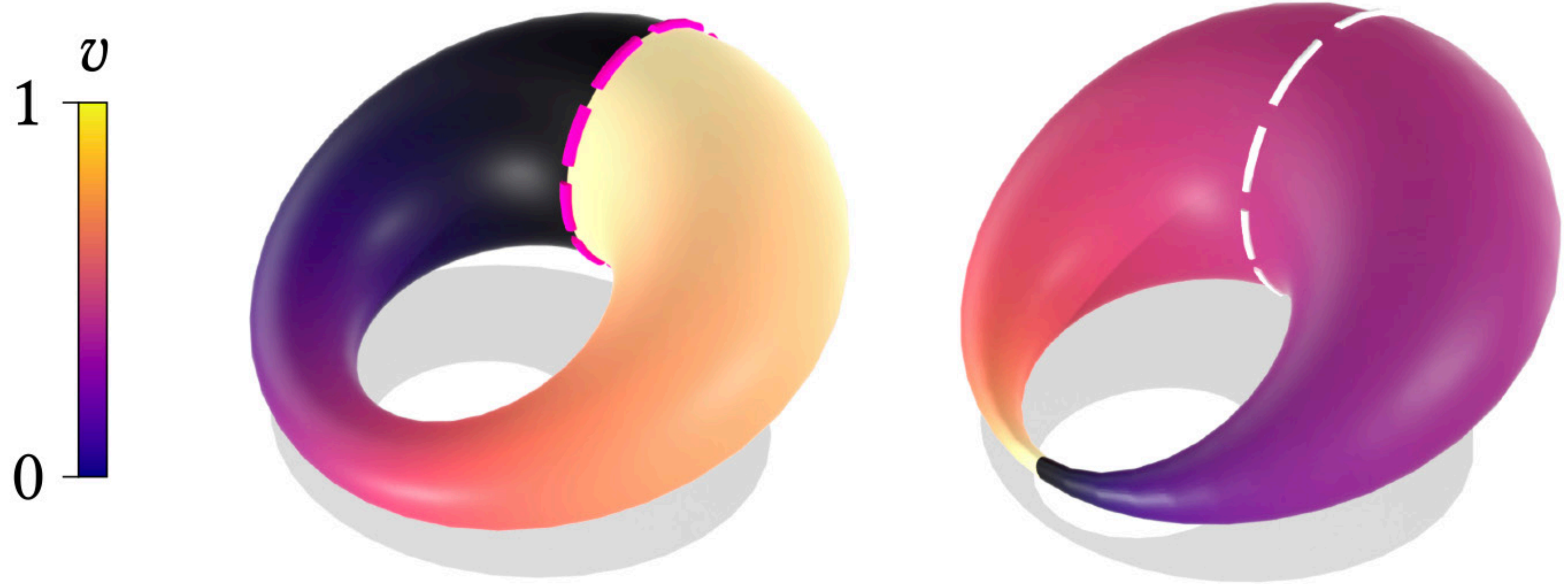
Recall objective function: $\min_{v:\, M \to \mathbb{R}} \int |\text{the jumps not across } \Gamma| + \varepsilon \int |\text{the jumps across } \Gamma|$

encourage jumps across Γ



Unexpected nonbounding loops

Recall objective function: $\min_{v:\, M\to\mathbb{R}} \int |\text{the jumps not across } \Gamma| + \boxed{\varepsilon \int |\text{the jumps across } \Gamma|}$ *encourage jumps across Γ*



Can always adversarially increase a handle's taper.

Unexpected nonbounding loops

Recall objective function: $\min_{v:\, M \to \mathbb{R}} \int |\text{the jumps not across } \Gamma| + \varepsilon \int |\text{the jumps across } \Gamma|$

encourage jumps across Γ

Unexpected nonbounding loops

Recall objective function: $\min_{v:\, M \to \mathbb{R}} \int |\text{the jumps not across } \Gamma| + \varepsilon \int |\text{the jumps across } \Gamma|$

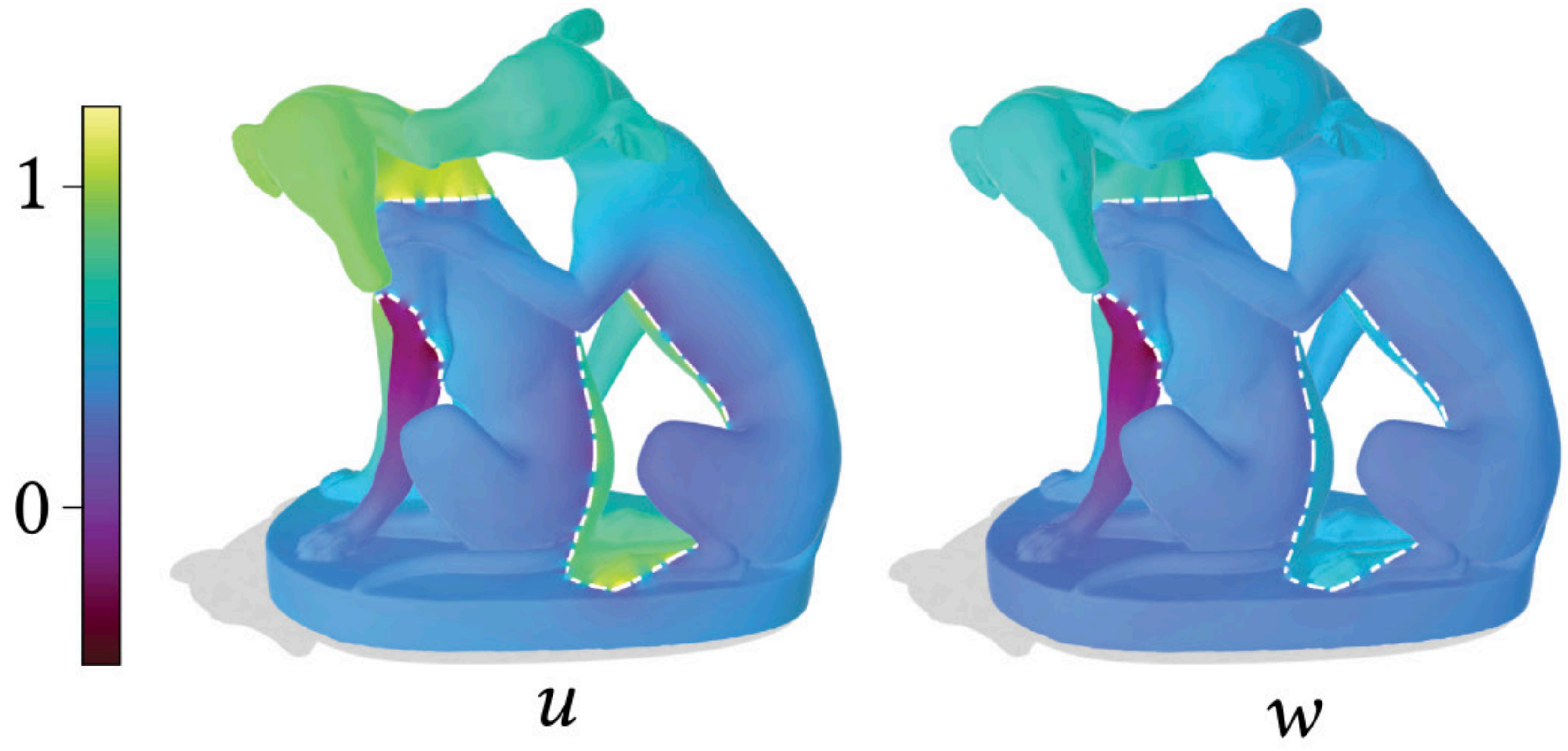
encourage jumps across Γ

But given a *fixed* mesh, we will recover the correct solution as gaps → 0 (and appropriate choice of ε).

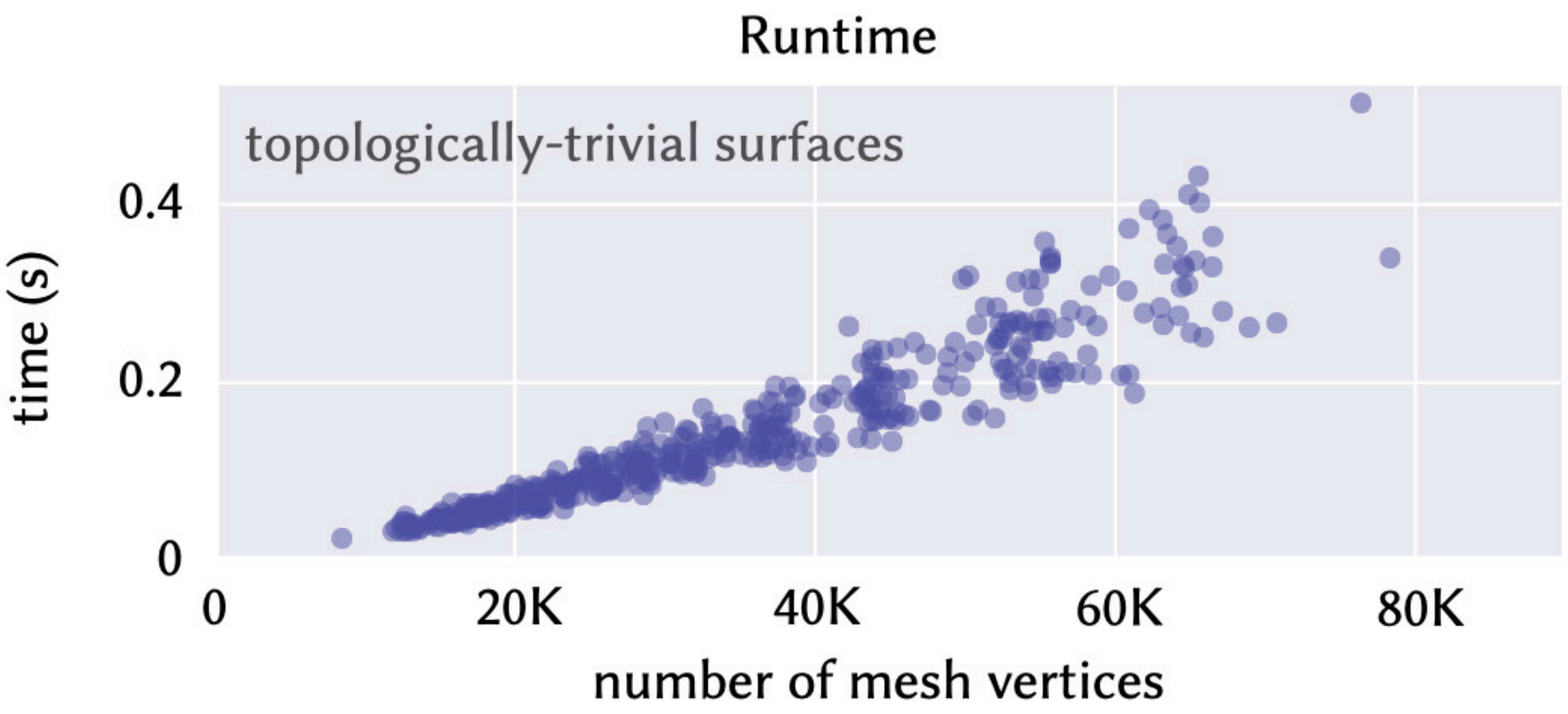
Unexpected nonbounding loops

Recall objective function: $\min_{v:\, M \to \mathbb{R}} \int |\text{the jumps not across } \Gamma| + \varepsilon \int |\text{the jumps across } \Gamma|$

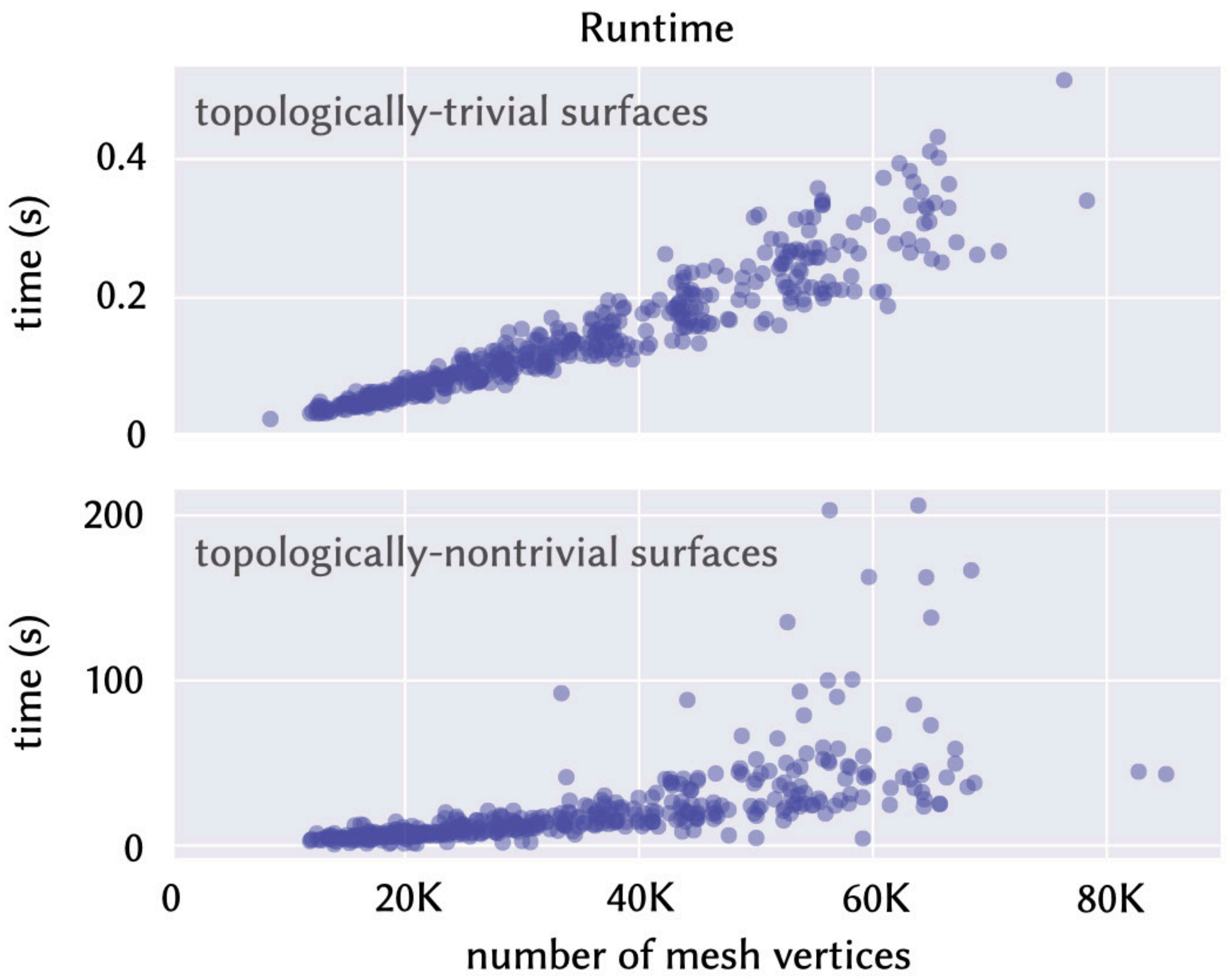
encourage jumps across Γ



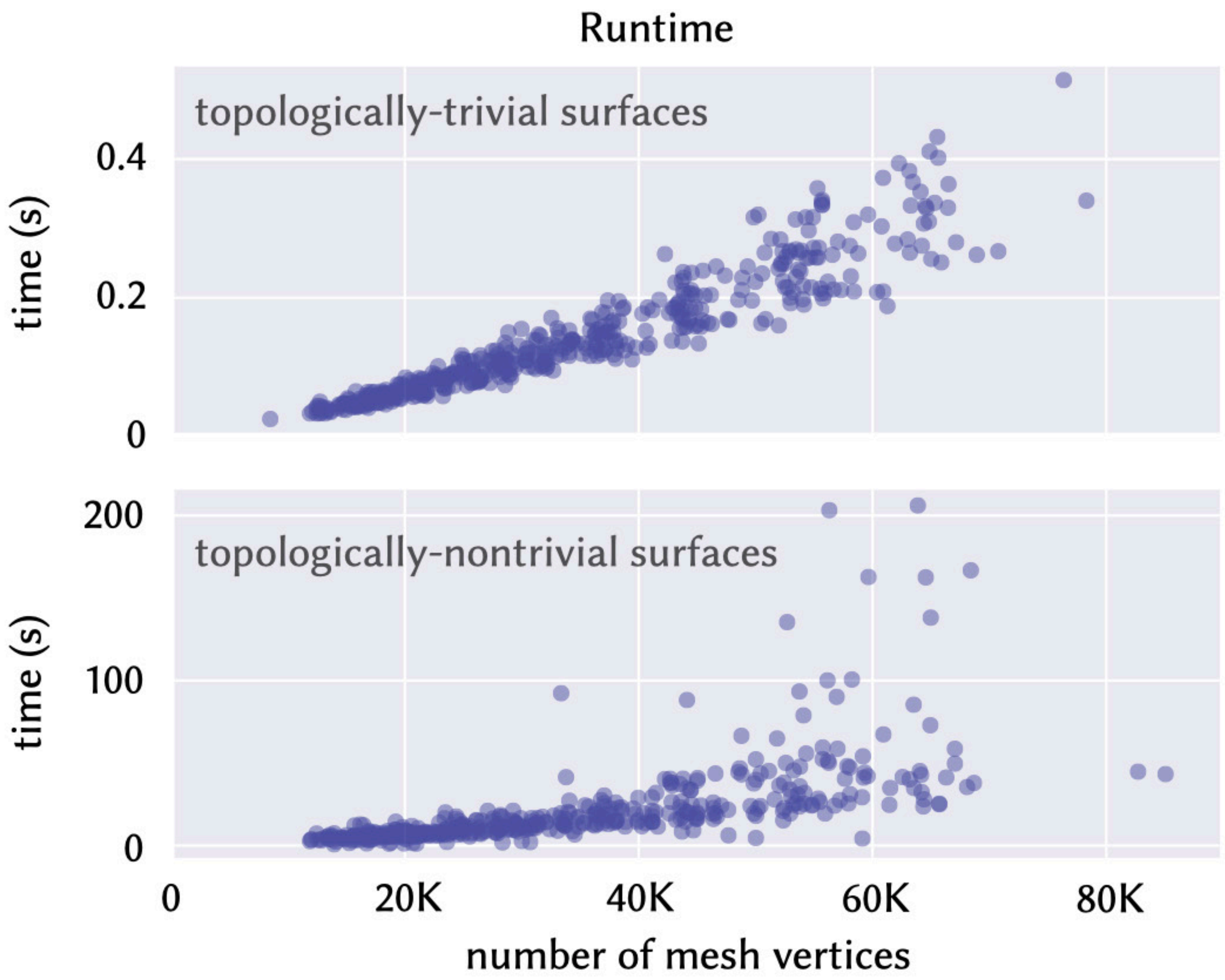
Performance



Performance



Performance



Computation dominated by linear program.

Reduced-size linear program

Current implementation simultaneously optimizes both jump locus *and* jump magnitudes.

Reduced-size linear program

Current implementation simultaneously optimizes both jump locus *and* jump magnitudes.

Instead:

Reduced-size linear program

Current implementation simultaneously optimizes both jump locus *and* jump magnitudes.

Instead:

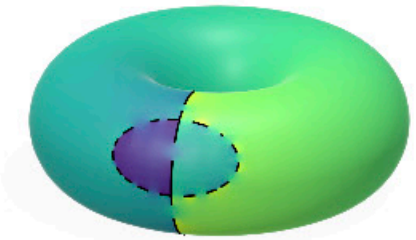
1. Use a *separate* shortest-path heuristic (Dijkstra) to complete Γ.

Reduced-size linear program

Current implementation simultaneously optimizes both jump locus *and* jump magnitudes.

Instead:

1. Use a *separate* shortest-path heuristic (Dijkstra) to complete Γ.



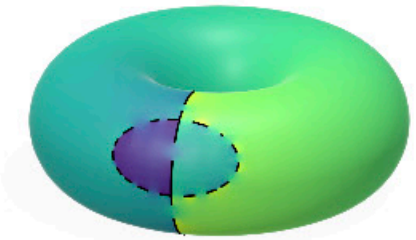
u

Reduced-size linear program

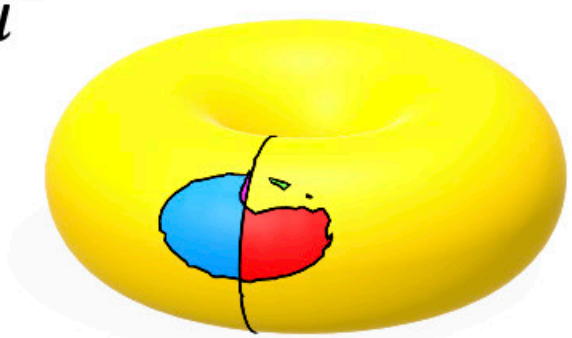
Current implementation simultaneously optimizes both jump locus *and* jump magnitudes.

Instead:

1. Use a *separate* shortest-path heuristic (Dijkstra) to complete Γ.



u



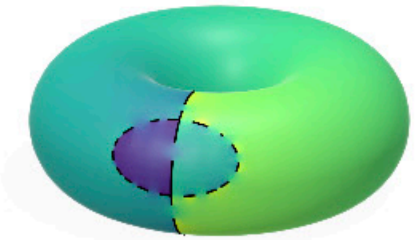
Dijkstra-completed curve

Reduced-size linear program

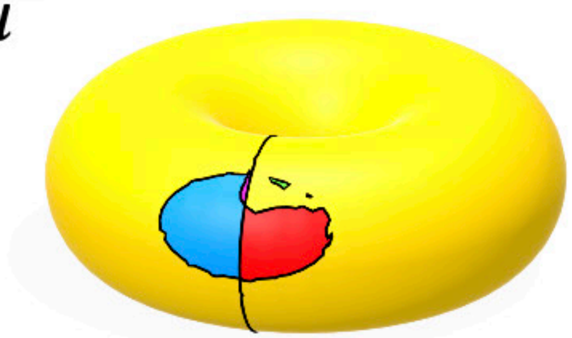
Current implementation simultaneously optimizes both jump locus *and* jump magnitudes.

Instead:

1. Use a *separate* shortest-path heuristic (Dijkstra) to complete Γ.

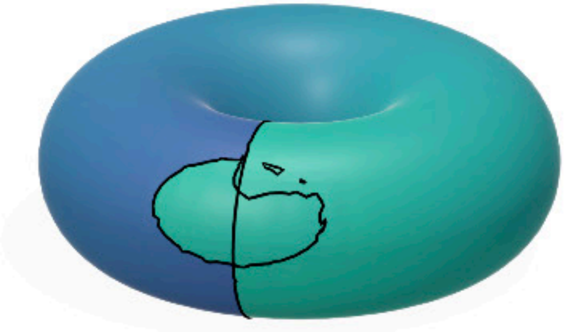


u



Dijkstra-completed curve

Integrate γ locally within each connected component

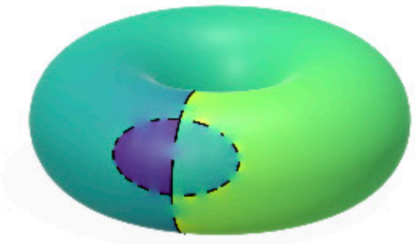


Reduced-size linear program

Current implementation simultaneously optimizes both jump locus *and* jump magnitudes.

Instead:

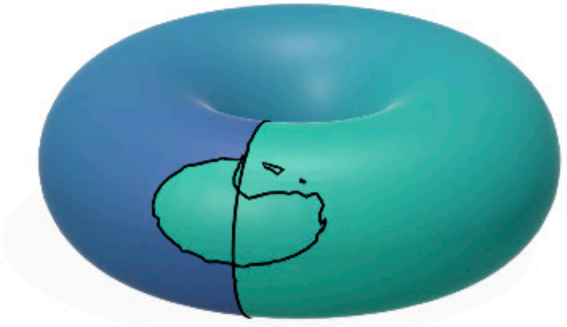
1. Use a *separate* shortest-path heuristic (Dijkstra) to complete Γ.



u

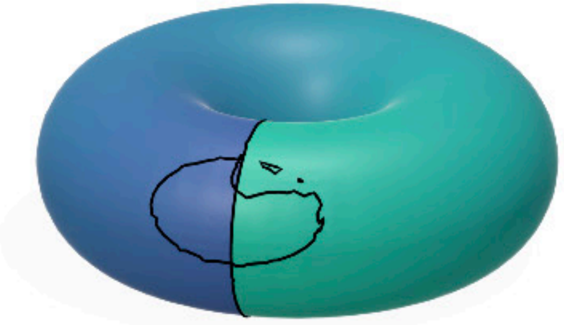
Dijkstra-completed curve

Integrate γ locally within each connected component



2. Minimize the L^1 norm of jumps across connected components.

v



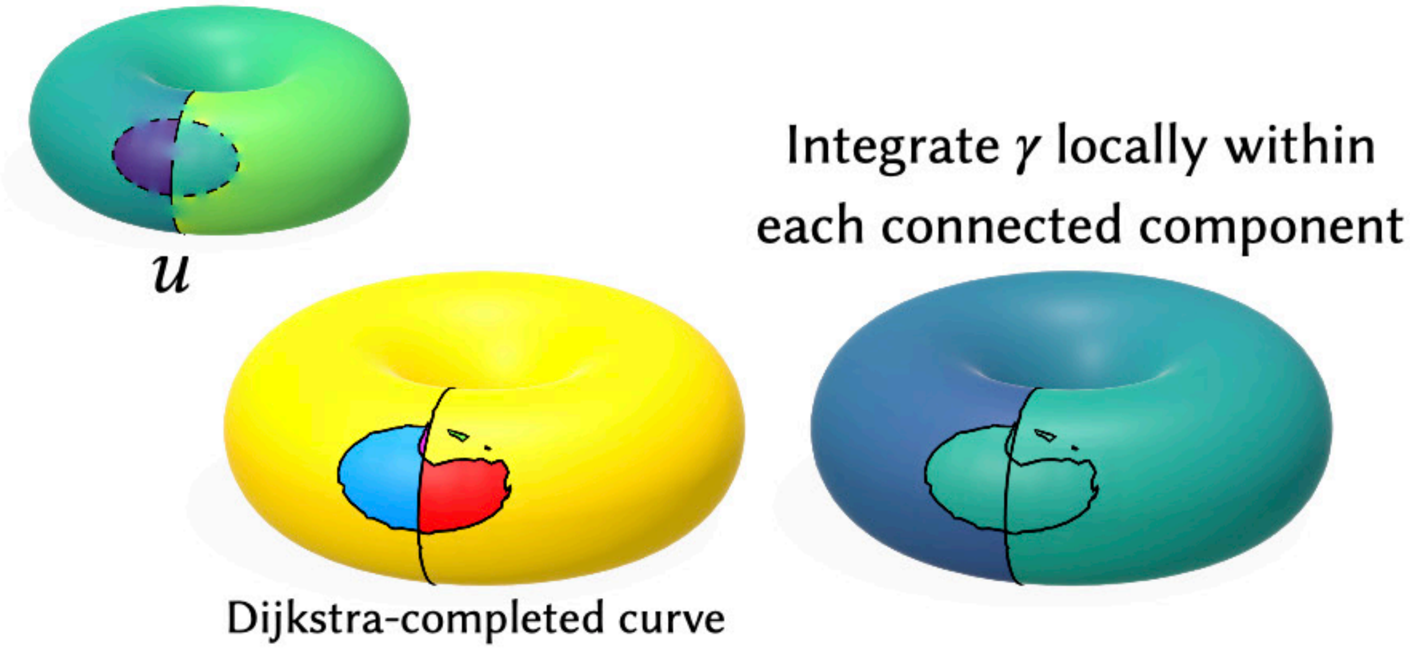
Reduced-size linear program

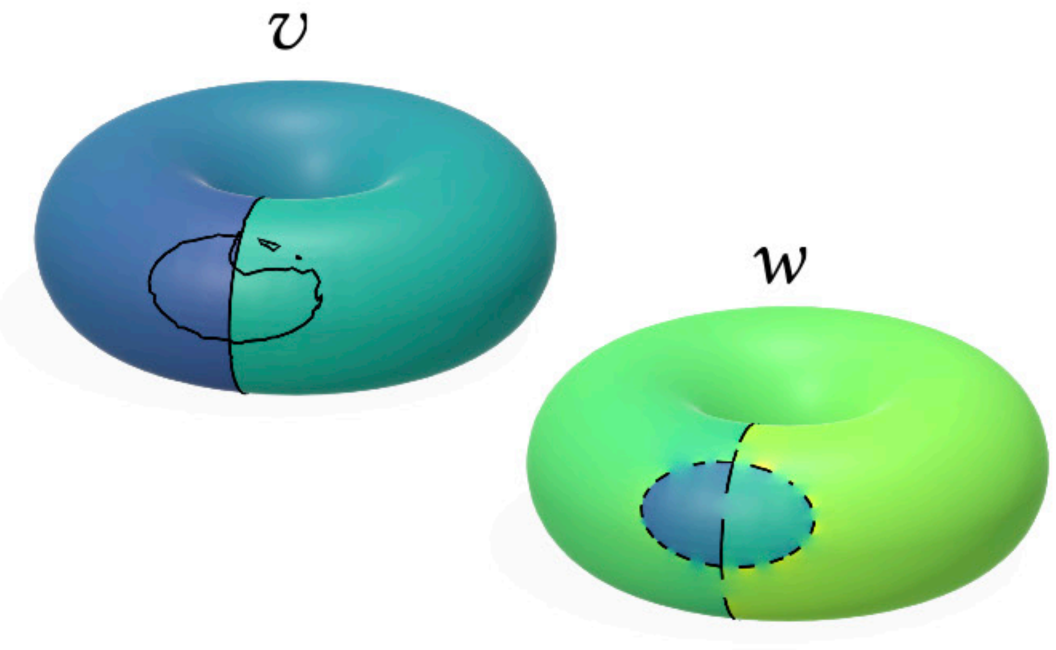
Current implementation simultaneously optimizes both jump locus *and* jump magnitudes.

Instead:

1. Use a *separate* shortest-path heuristic (Dijkstra) to complete Γ.

2. Minimize the L^1 norm of jumps across connected components.





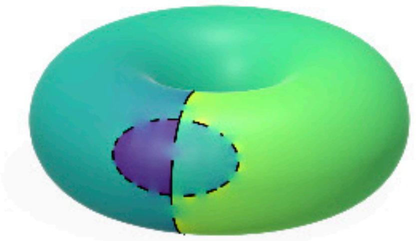
Reduced-size linear program

Current implementation simultaneously optimizes both jump locus *and* jump magnitudes.

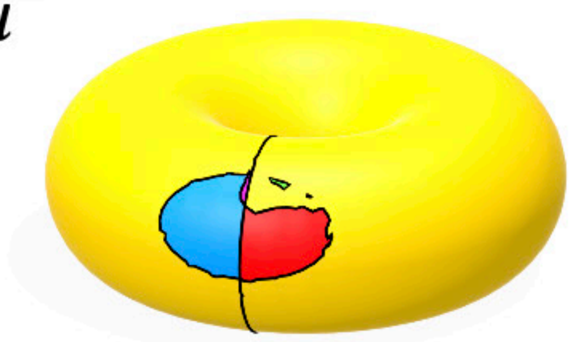
Instead:

1. Use a *separate* shortest-path heuristic (Dijkstra) to complete Γ.

2. Minimize the L^1 norm of jumps across connected components.

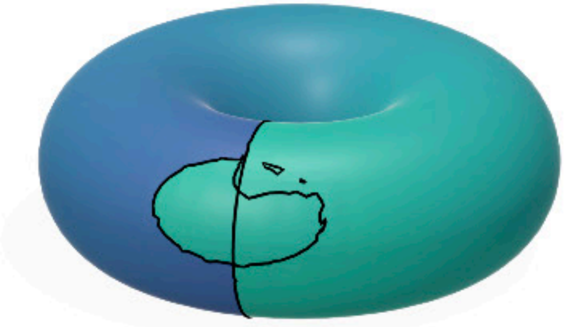


u

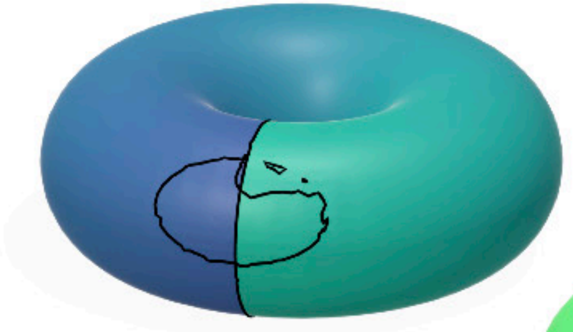


Dijkstra-completed curve

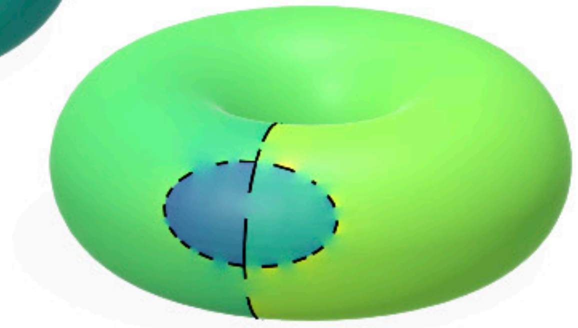
Integrate γ locally within each connected component



v

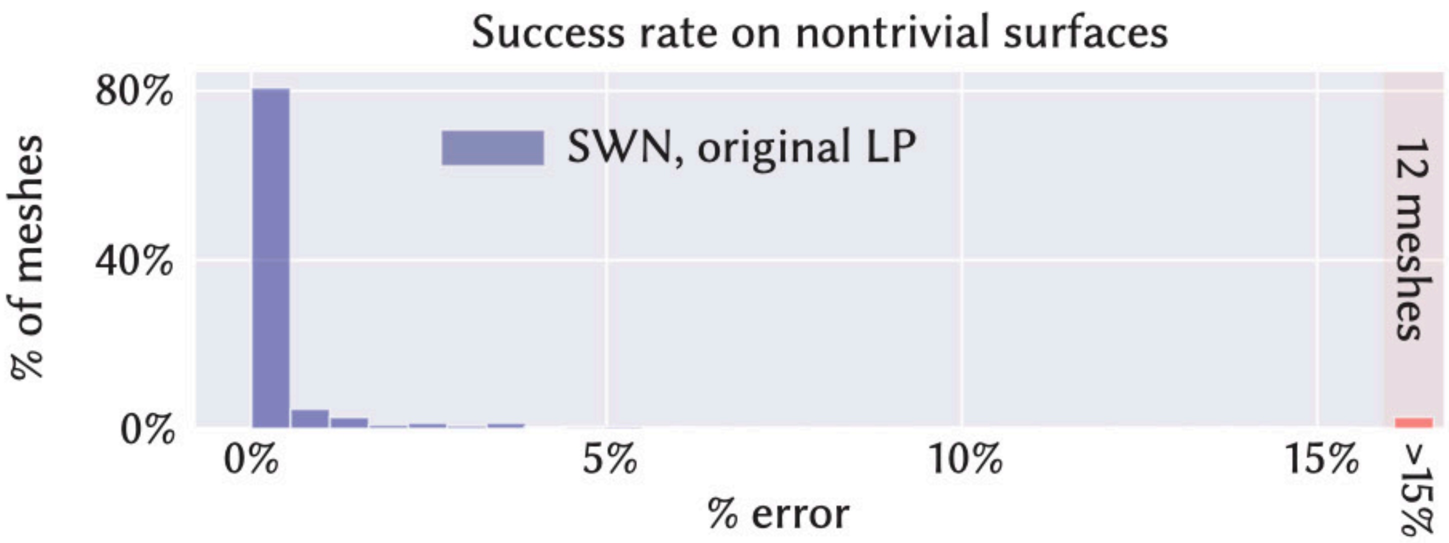


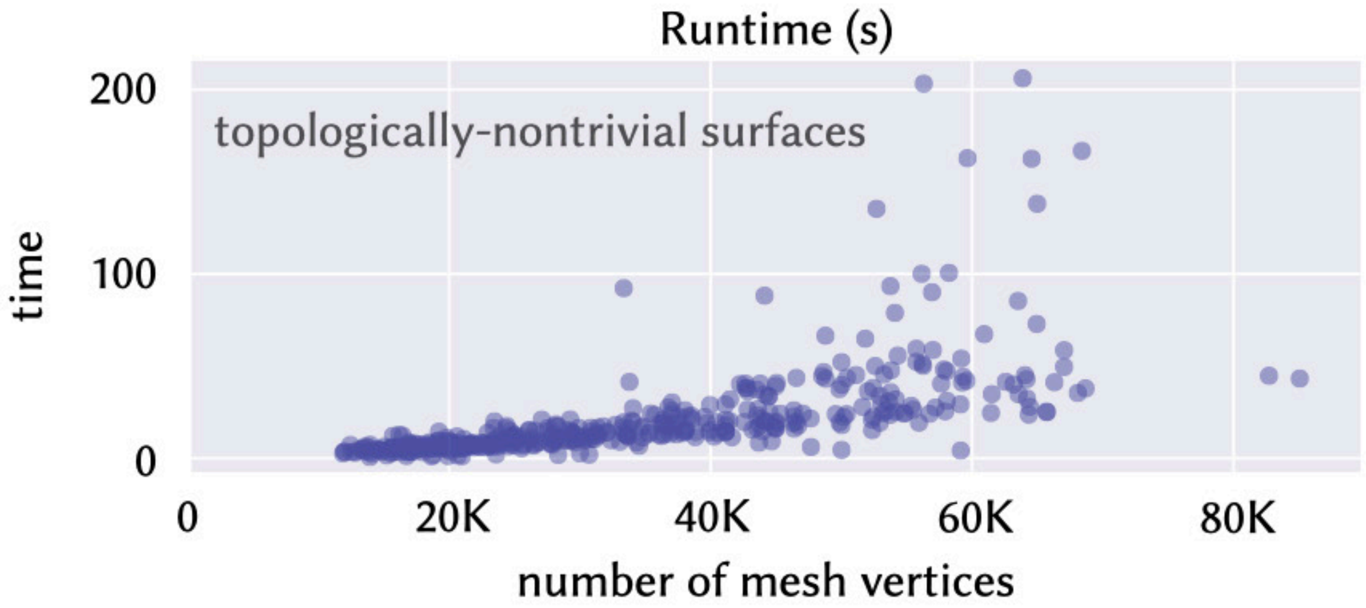
w



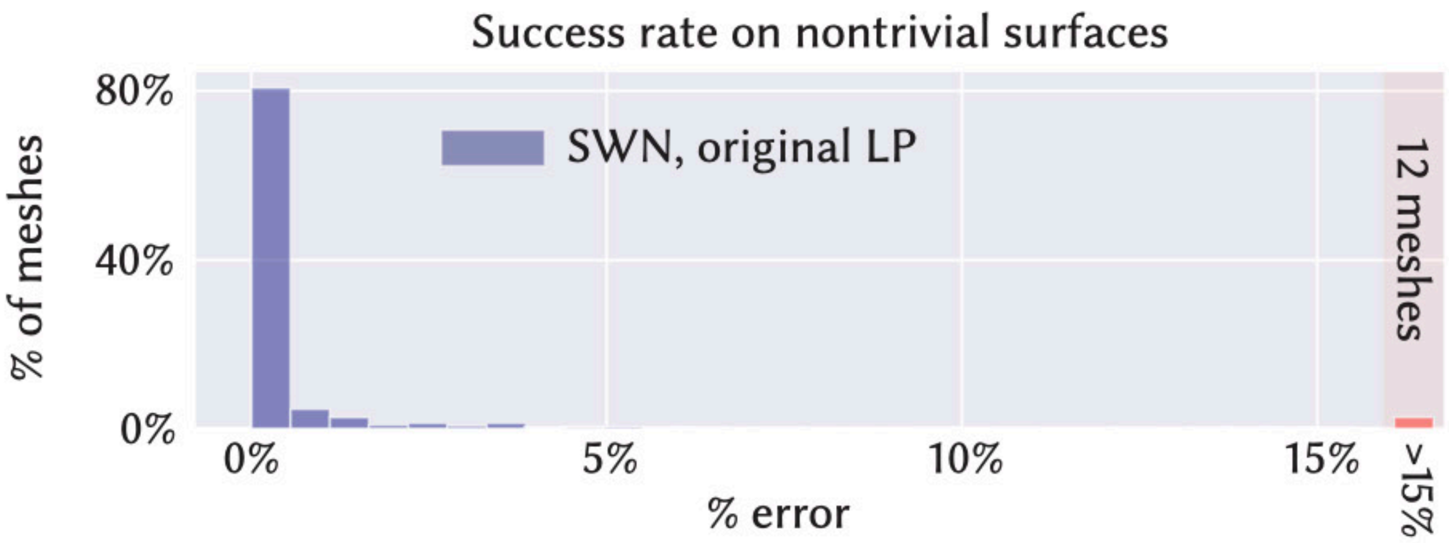
Number of DOFs: $|F| \rightarrow$ just a few connected components!

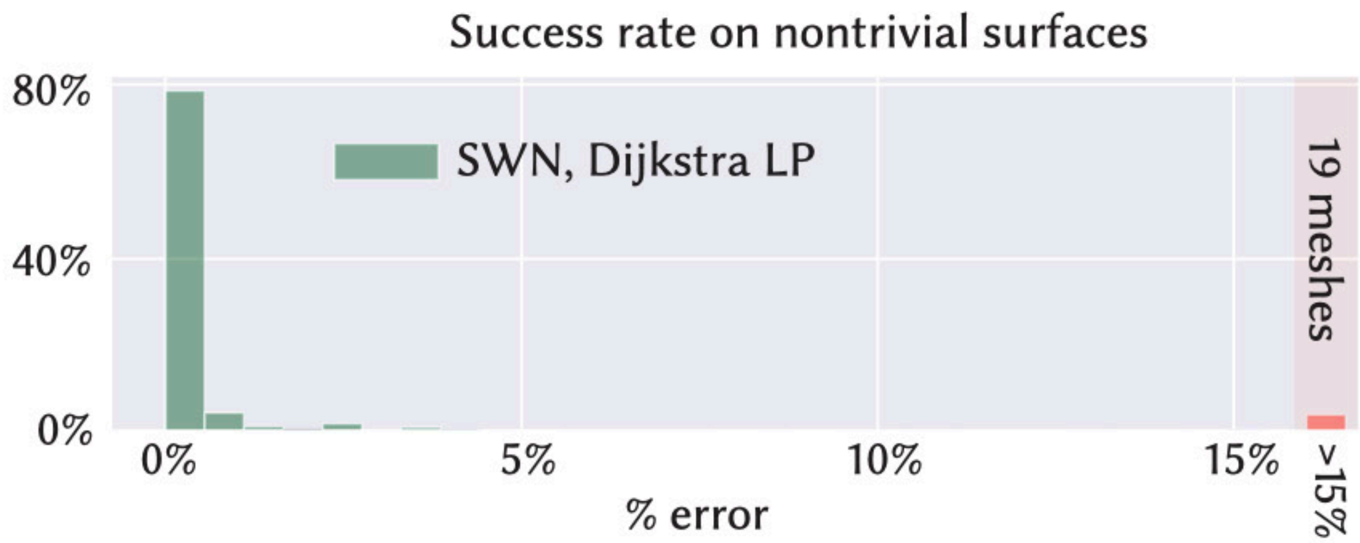
Approximate solutions, 100x faster

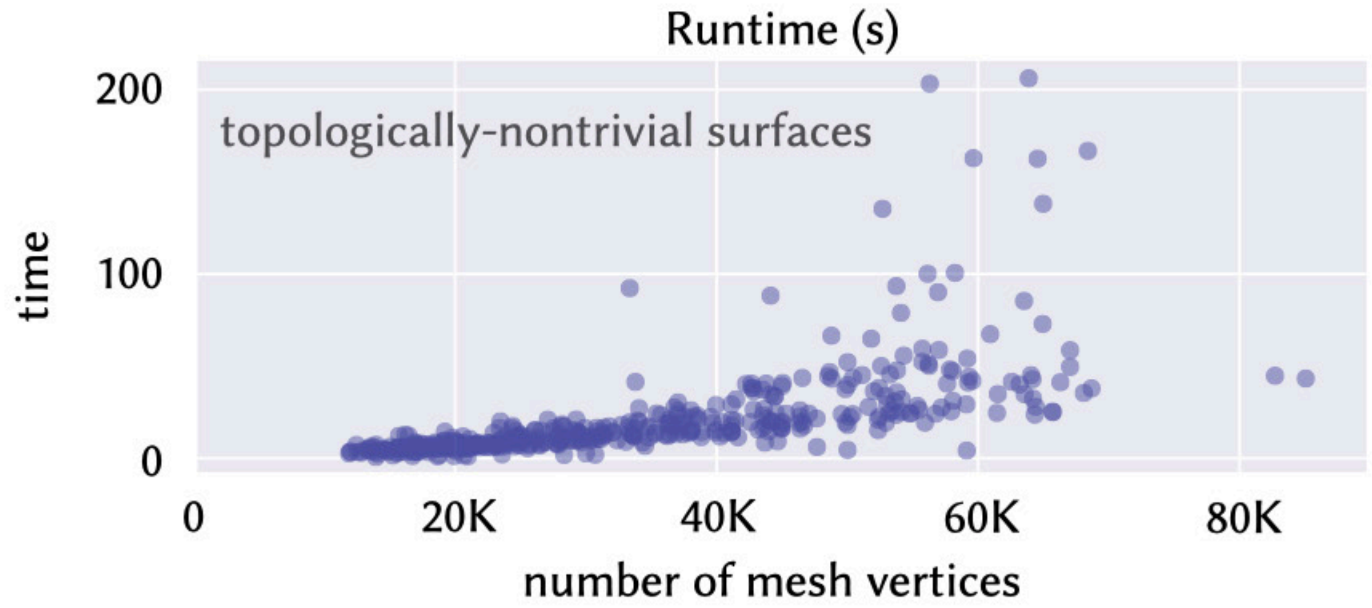




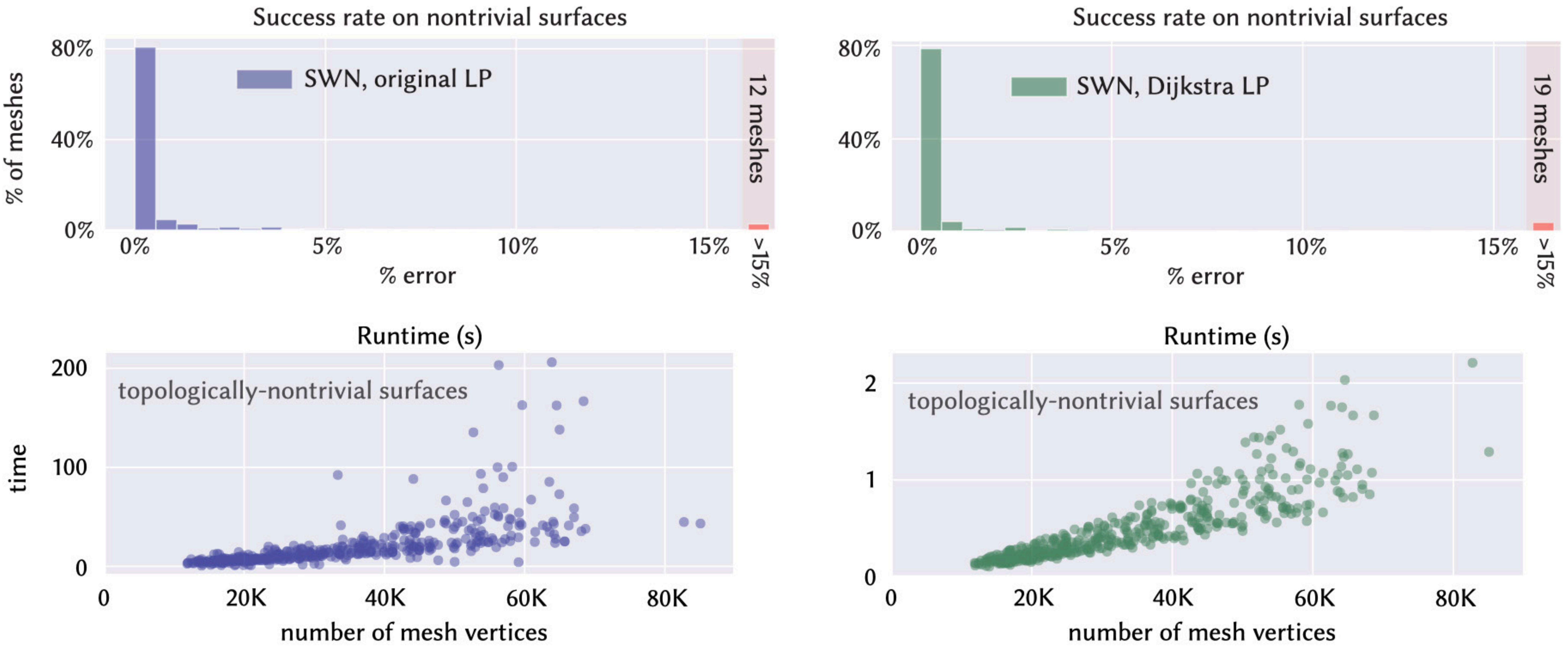
Approximate solutions, 100x faster



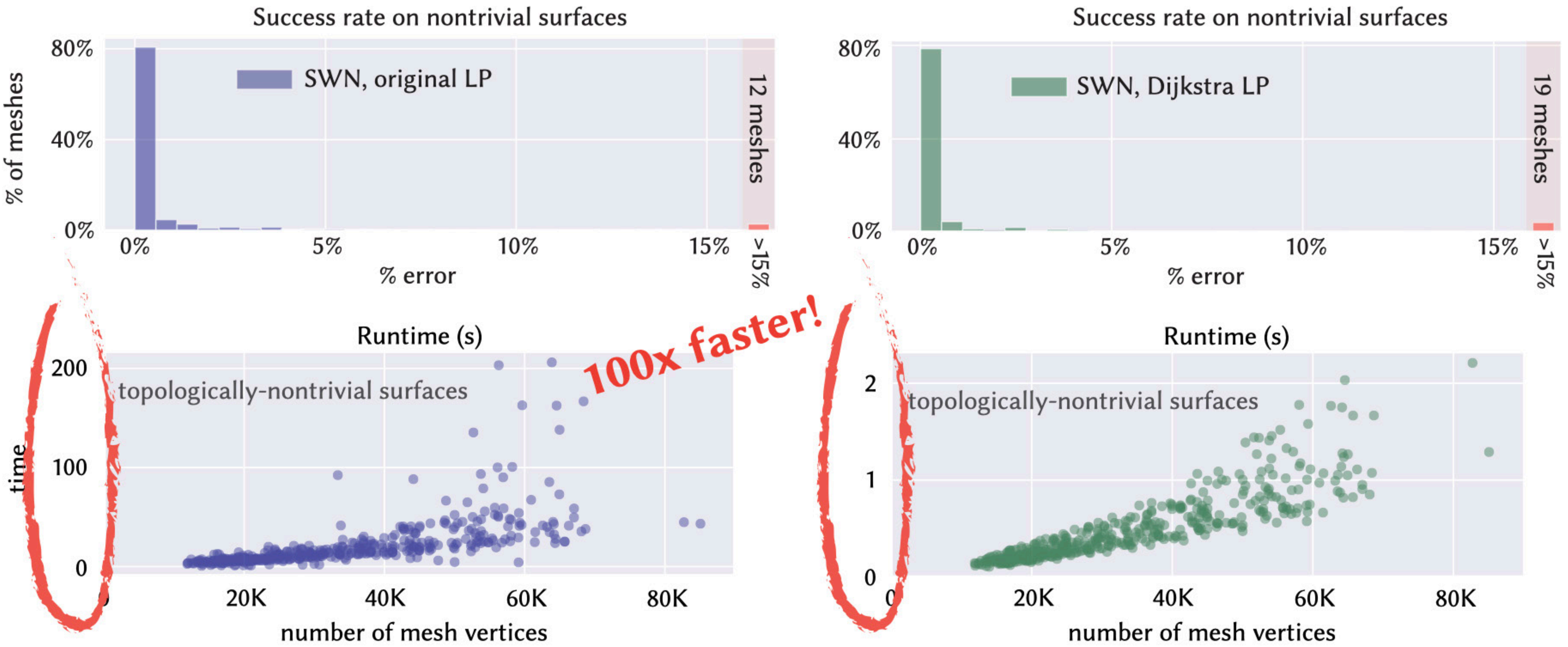




Approximate solutions, 100x faster



Approximate solutions, 100x faster



CONCLUSION

Takeaways

Takeaways

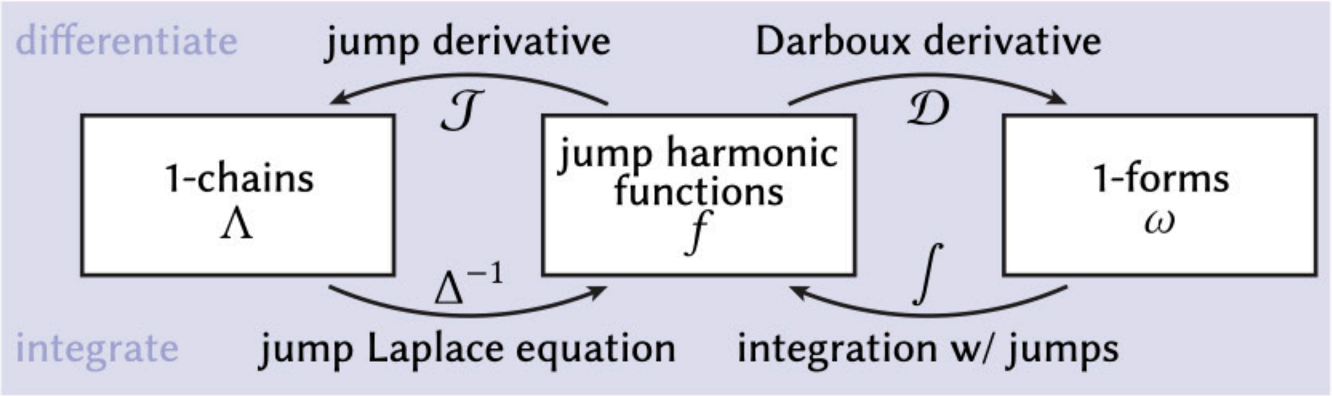
- Classic inside-outside definitions don't work on surfaces!

Takeaways

- Classic inside-outside definitions don't work on surfaces!
- Cohomology → robust homological geometry processing

Takeaways

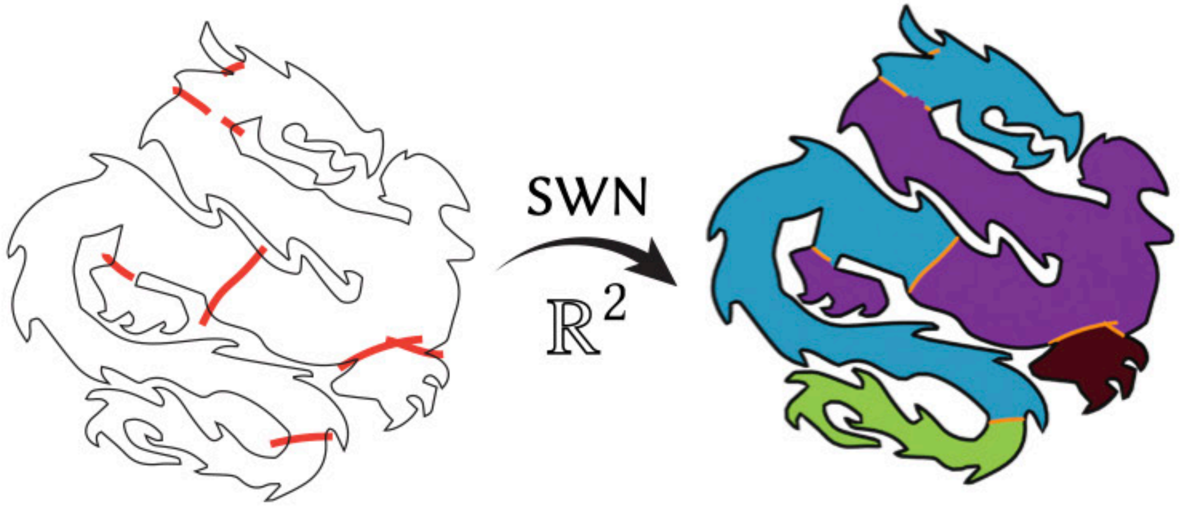
- Classic inside-outside definitions don't work on surfaces!
- Cohomology → robust homological geometry processing
  - Duality between curves and 1-forms → use jump harmonic functions to translate between the two



Fun future directions

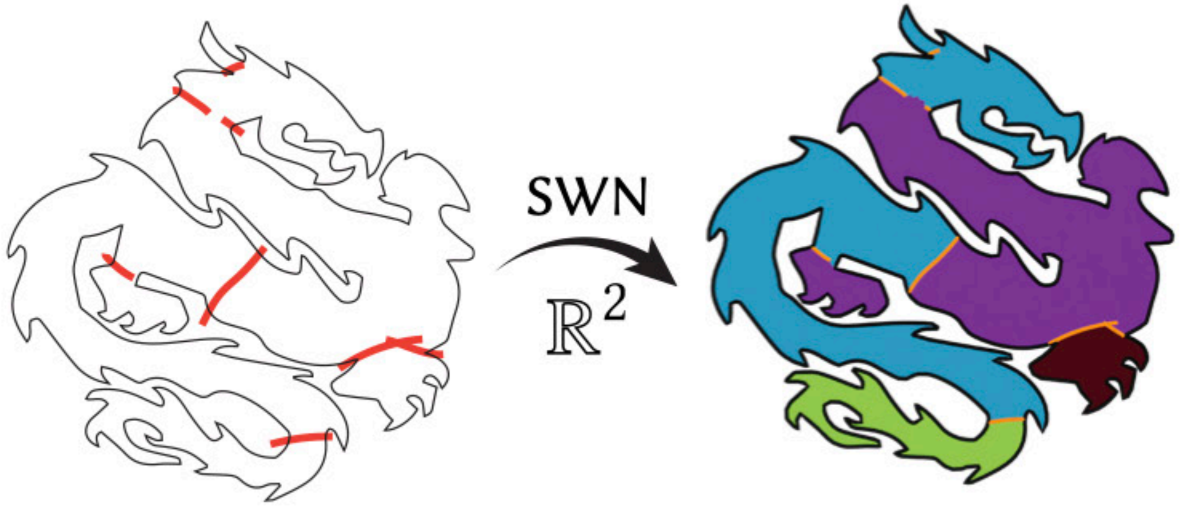
Fun future directions

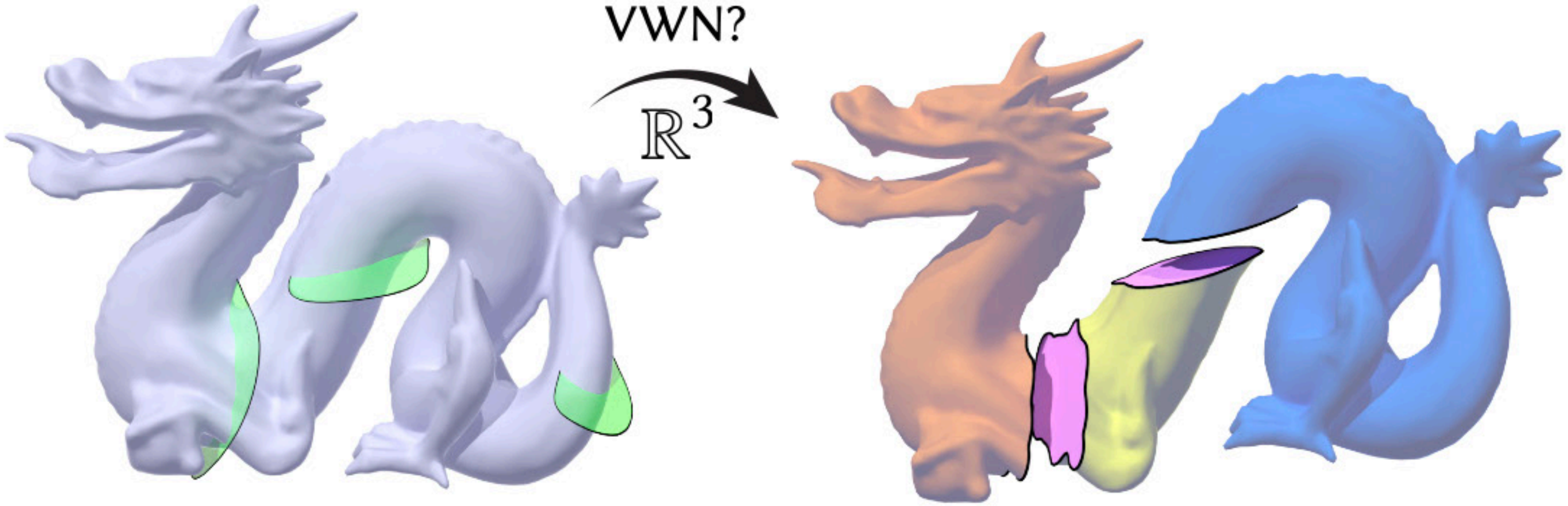
Subsets of $\mathbb{R}^n$:



Fun future directions

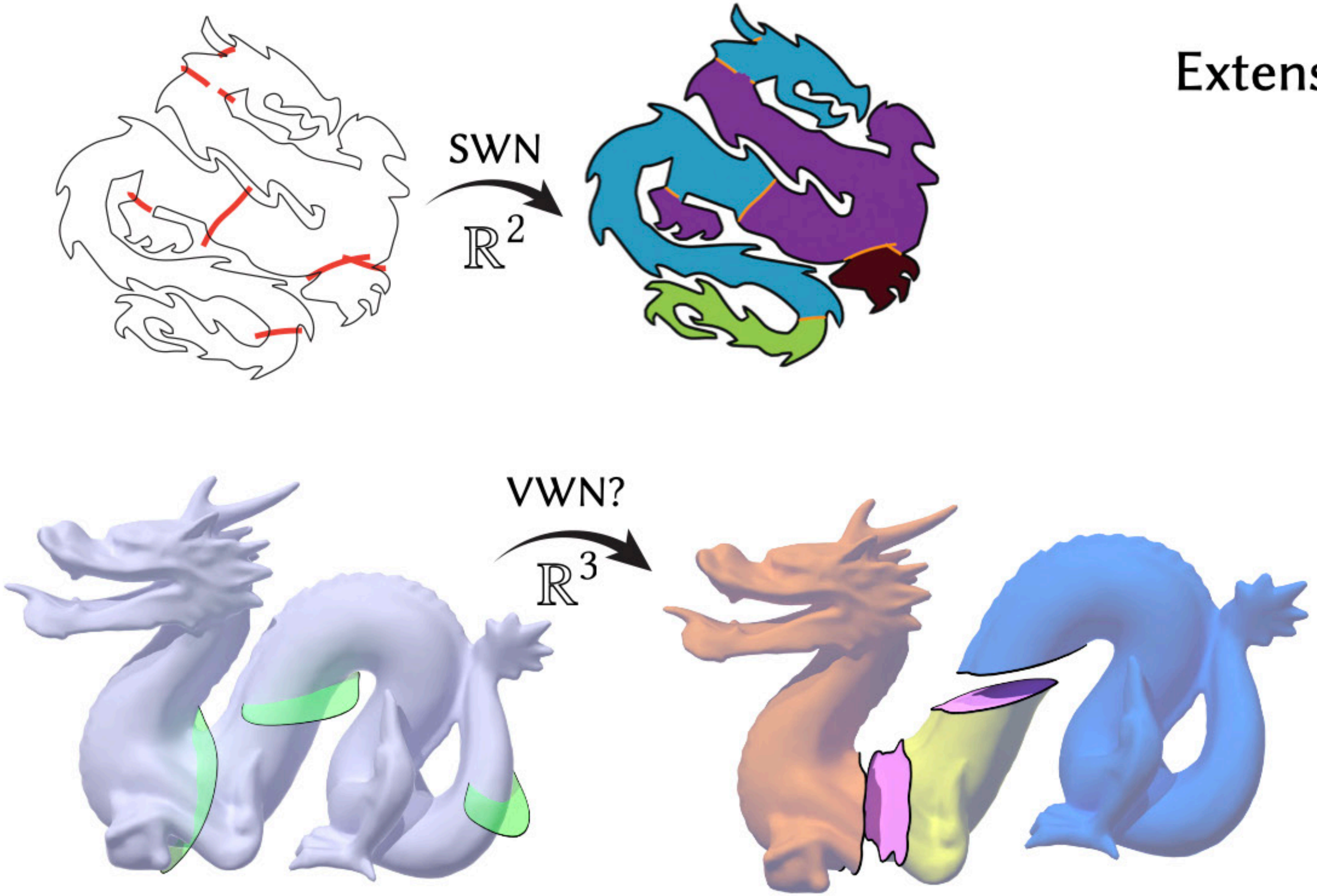
Subsets of $\mathbb{R}^n$:



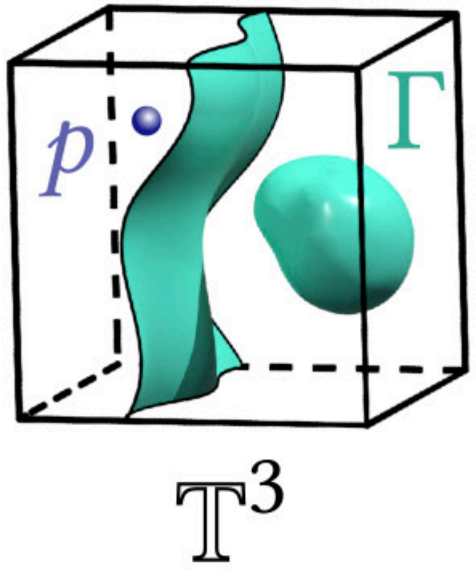


Fun future directions

Subsets of $\mathbb{R}^n$:

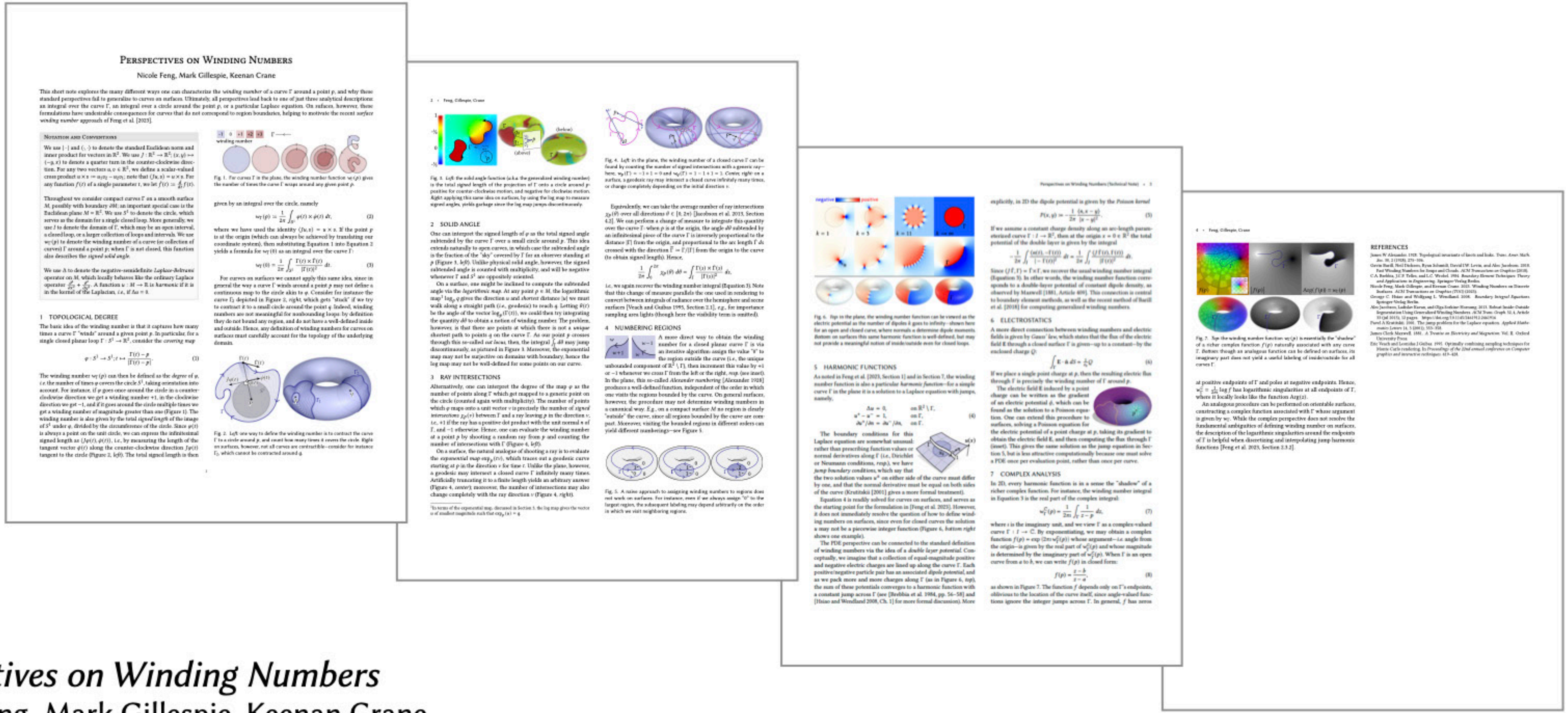


Extension of SWN to higher dimensions,
e.g. periodic domains in 3D.



Winding numbers are everywhere!

Many mathematical & physical interpretations — see our supplemental for details!



Perspectives on Winding Numbers
Nicole Feng, Mark Gillespie, Keenan Crane