



# SIGGRAPH 東京 ASIA 2024 TOKYO

Conference | 3–6 December 2024

Exhibition | 4–6 December 2024

Venue | Tokyo International Forum, Japan

## Differential Walk on Spheres

Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas

Sponsored by



Organized by





**Bailey Miller**  
CMU



**Rohan Sawhney**  
Nvidia



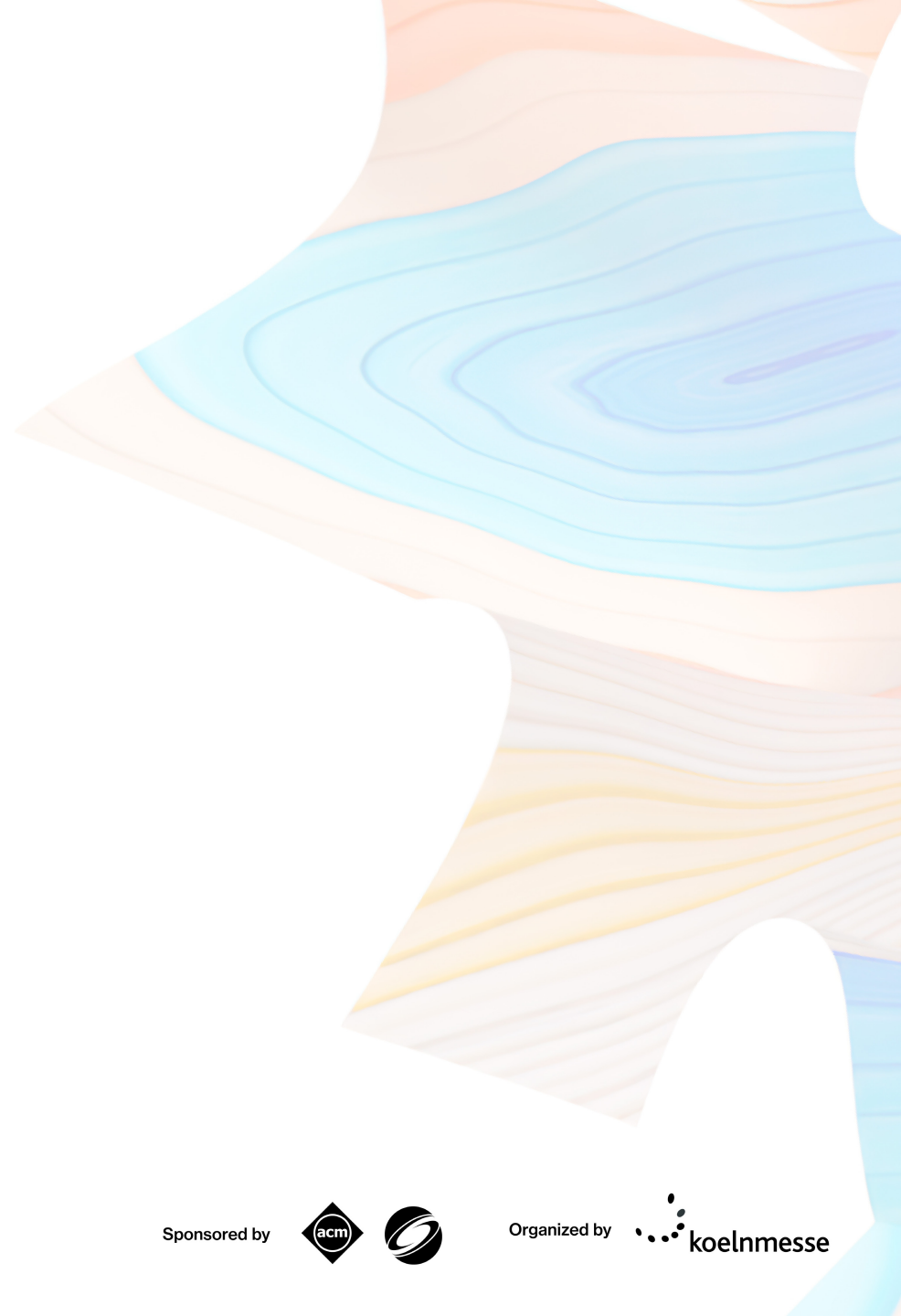
**Keenan Crane**  
CMU

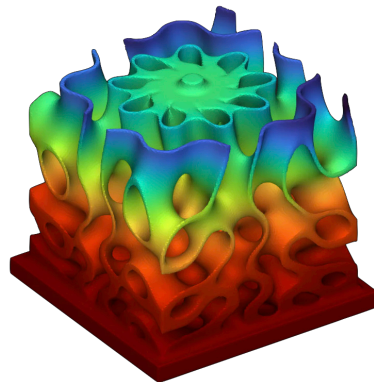


**Ioannis Gkioulekas**  
CMU

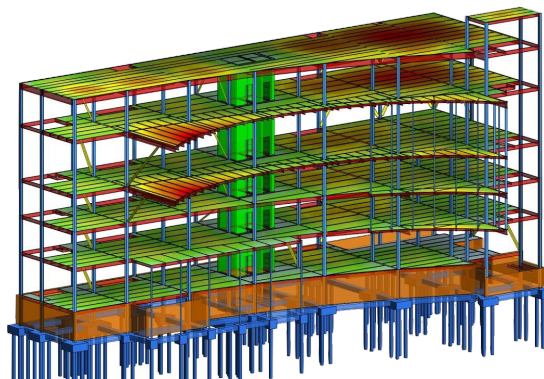
and support from







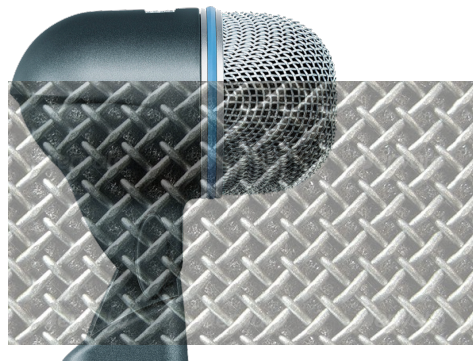
thermal diffusion



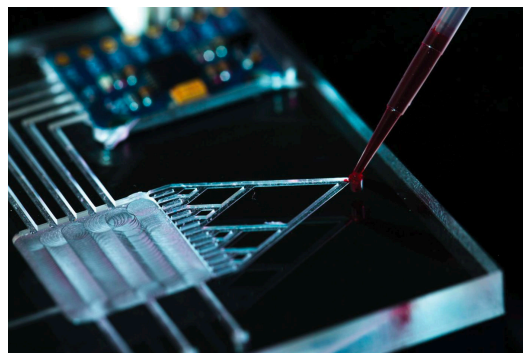
structural analysis



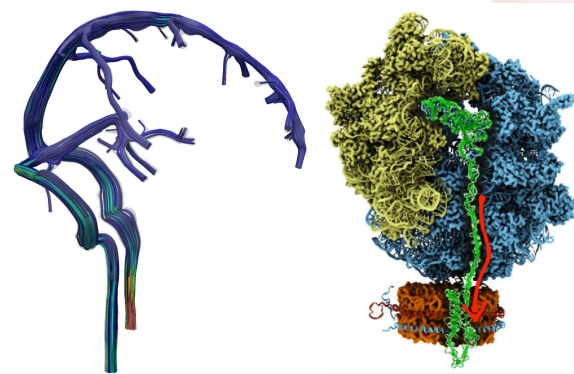
electrostatics



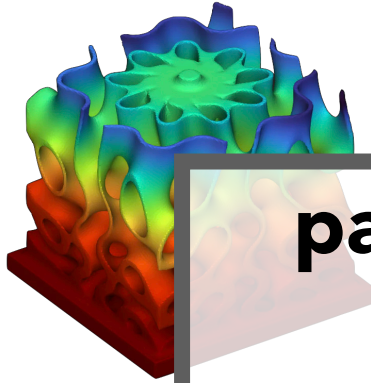
acoustic modeling



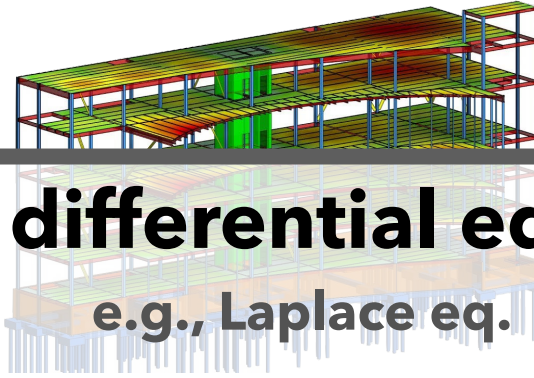
microfluidics



biophysics



thermal diffusion



structural analysis

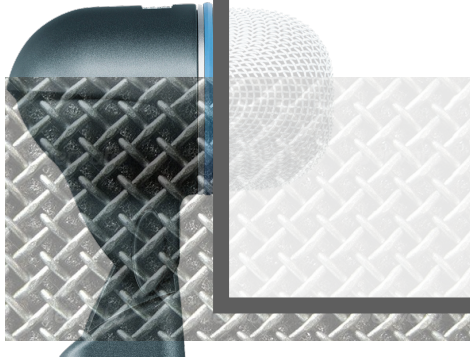


electrostatics

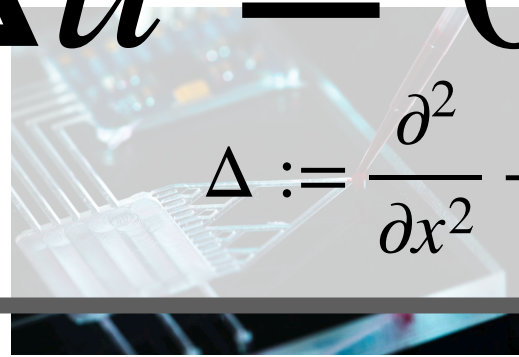
**partial differential equations,**  
e.g., Laplace eq.

$$\Delta u = 0$$

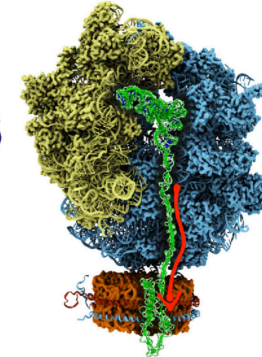
$$\Delta := \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$



acoustic modeling



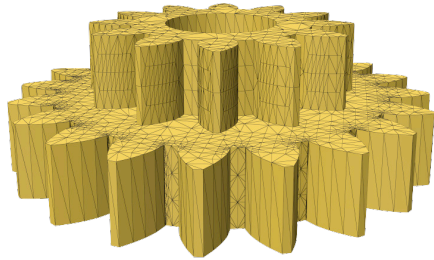
microfluidics



biophysics

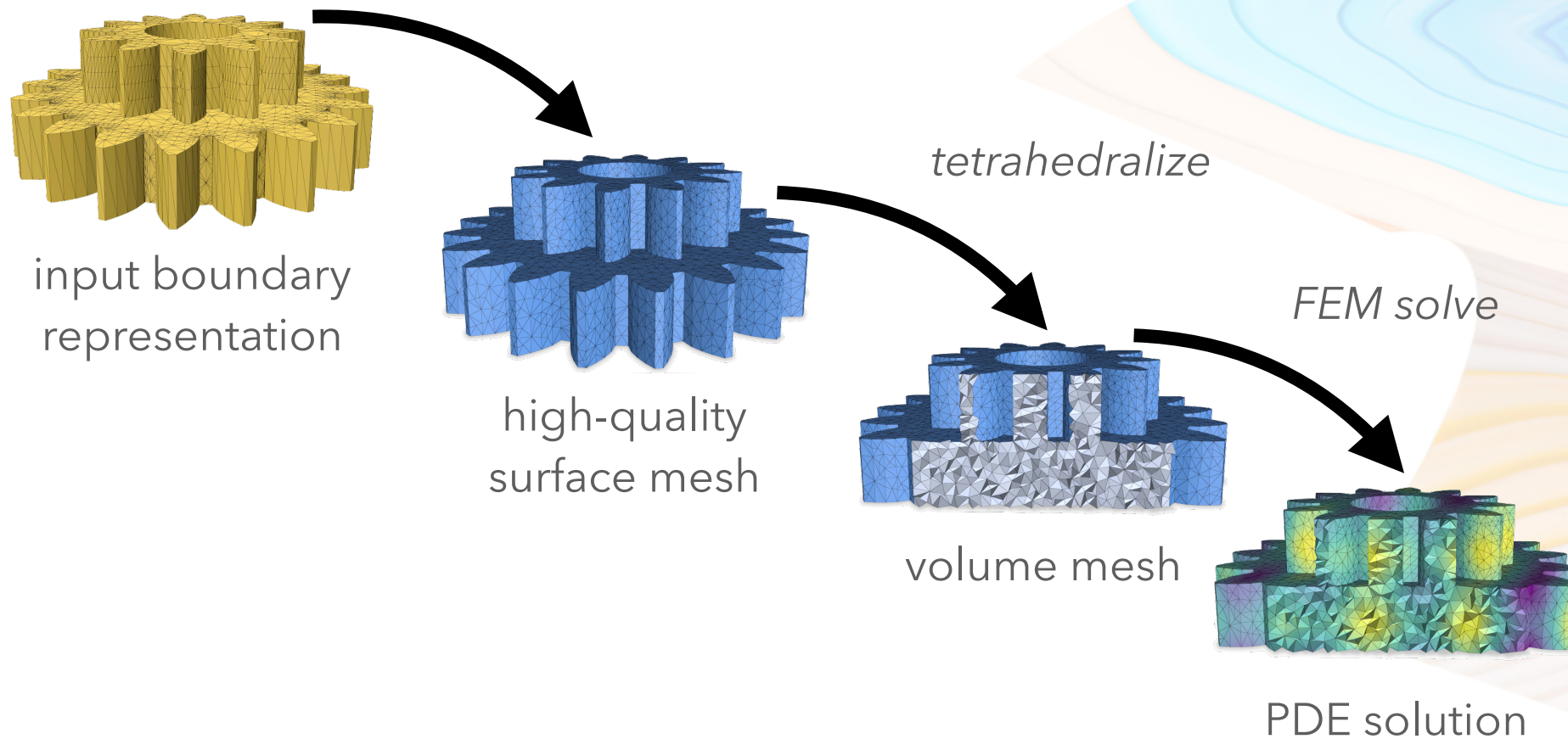


# finite element method (FEM) pipeline

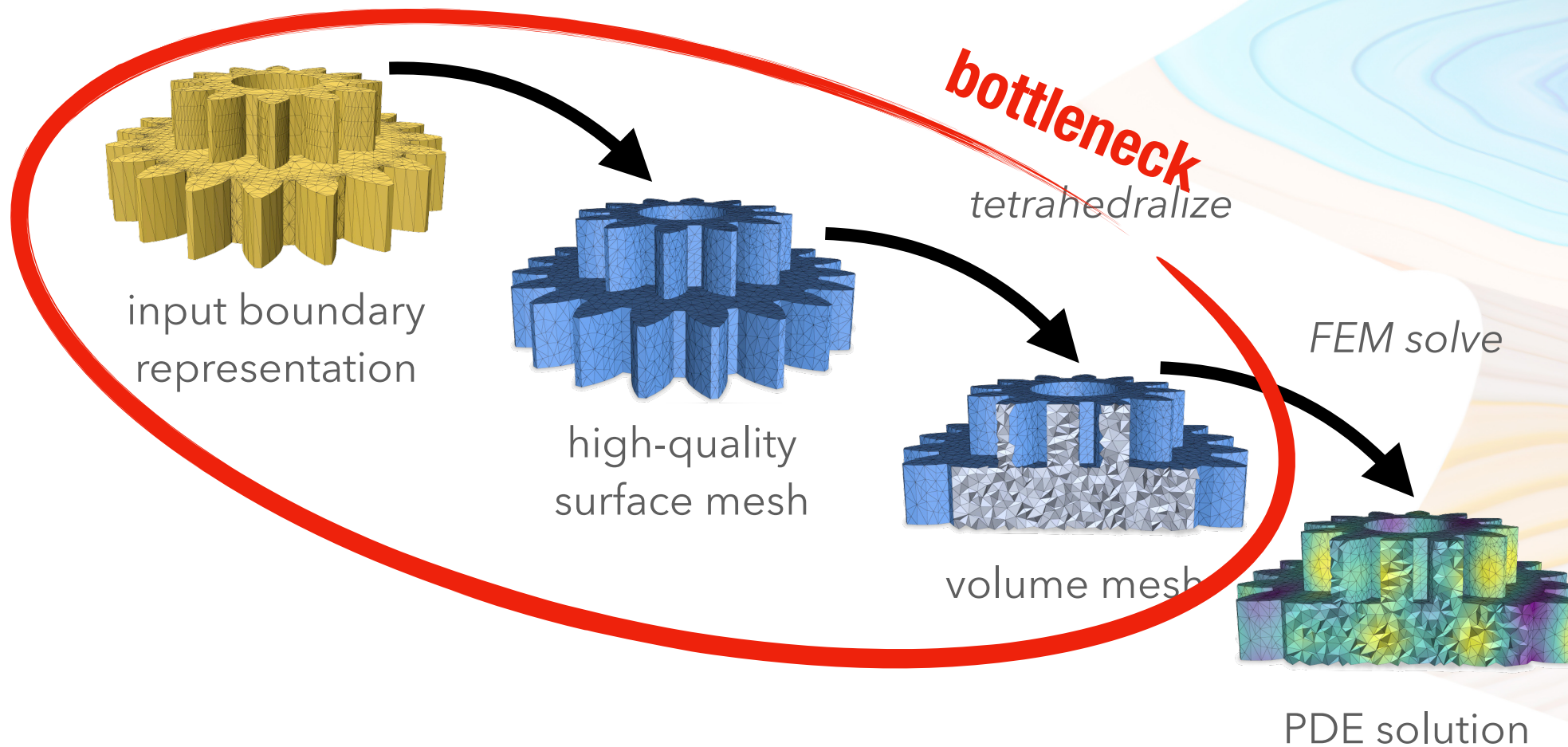


input boundary  
representation

# finite element method (FEM) pipeline



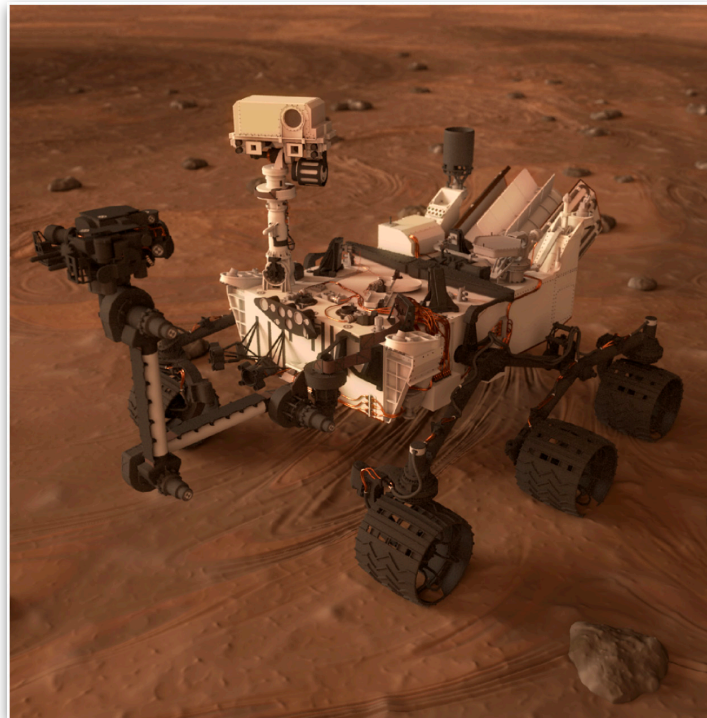
# finite element method (FEM) pipeline



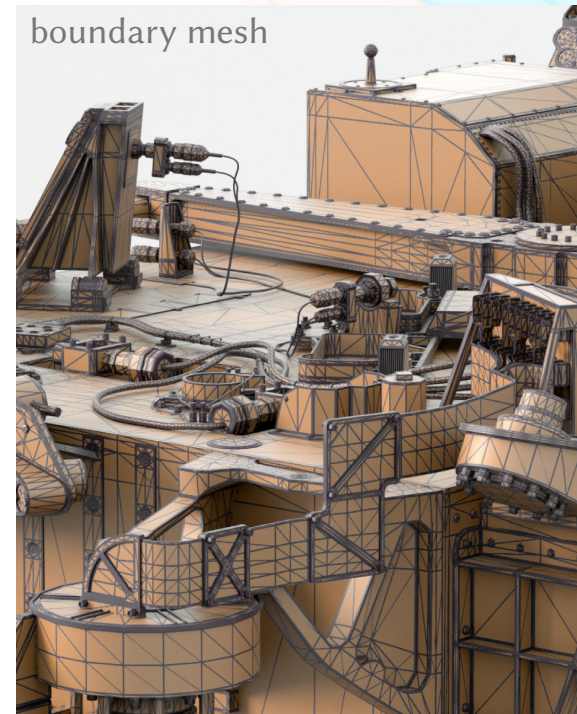


# meshing complex geometry is difficult + slow

Mars curiosity rover



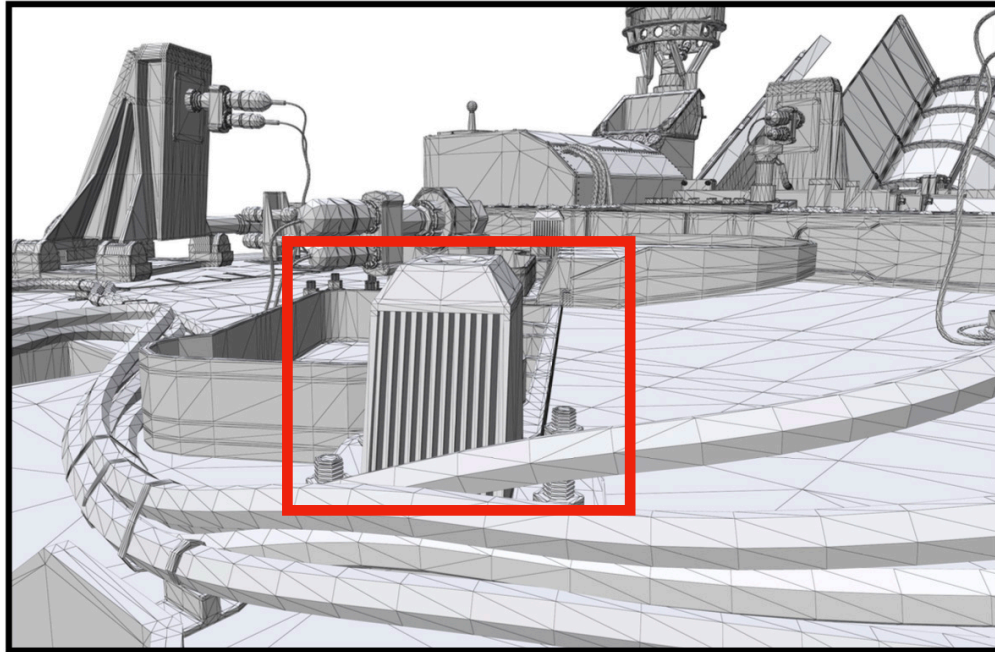
boundary mesh close up



[Miller et al. 2024]

# meshing complex geometry is difficult + slow

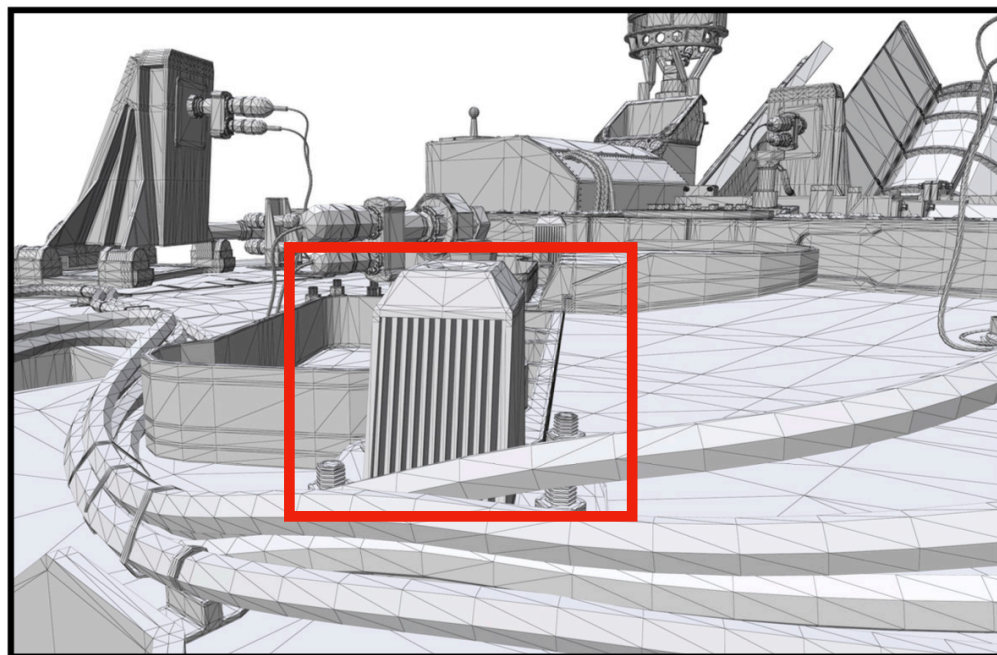
input boundary mesh



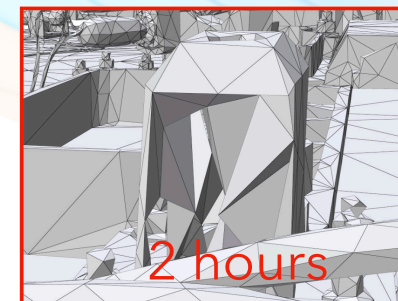
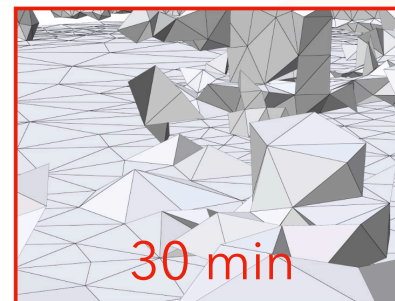
[Miller et al. 2024]

# meshing complex geometry is difficult + slow

input boundary mesh



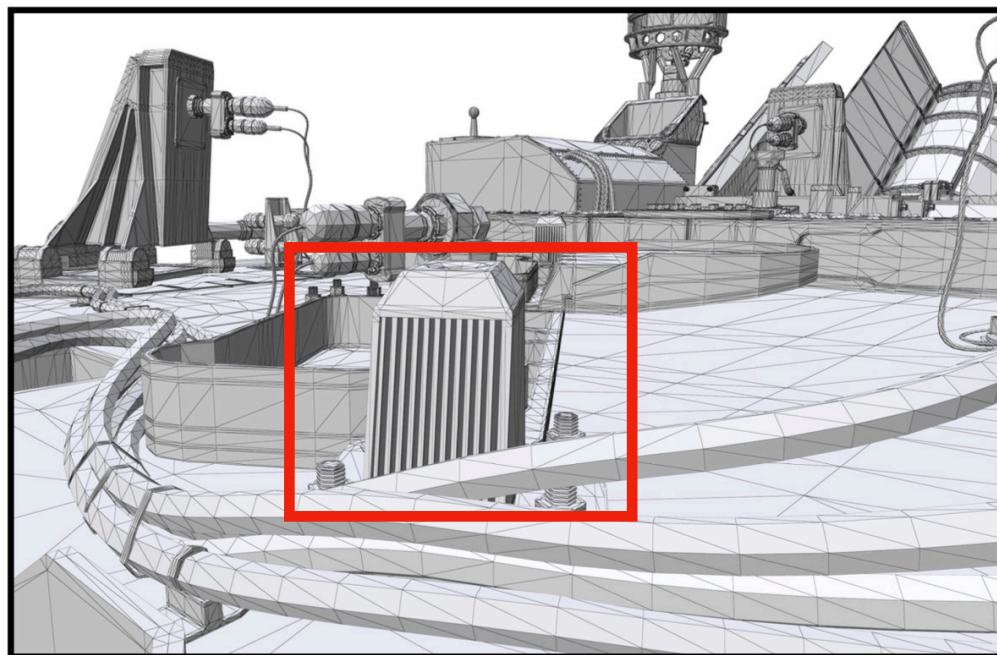
boundary of tetrahedral mesh



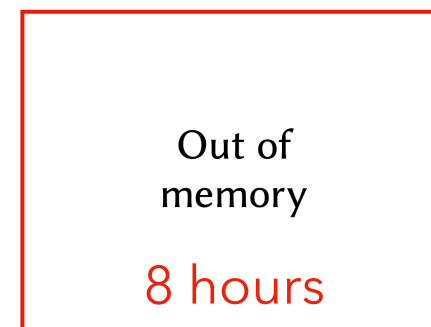
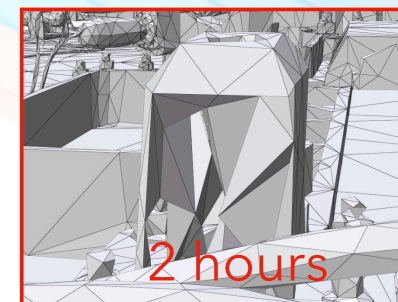
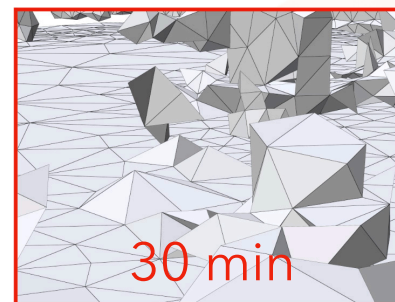
[Miller et al. 2024]

# meshing complex geometry is difficult + slow

input boundary mesh



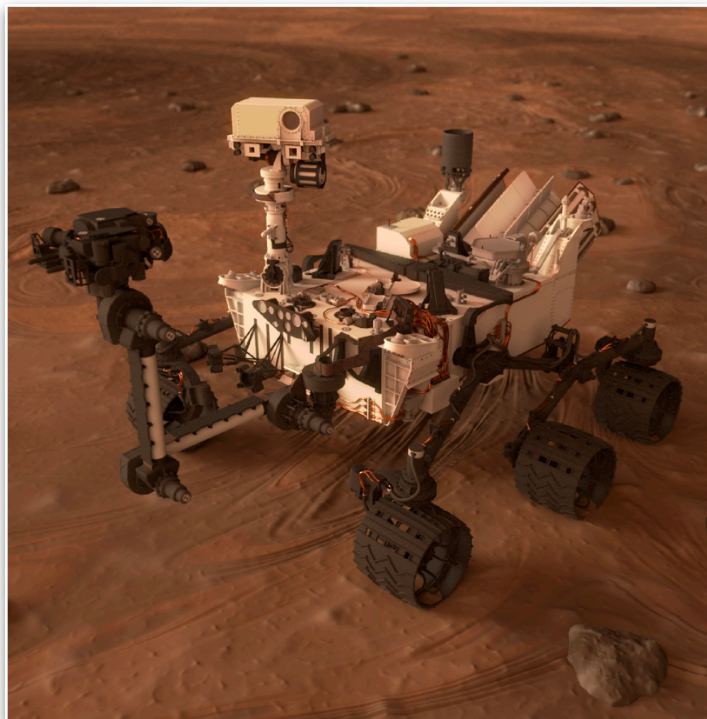
boundary of tetrahedral mesh



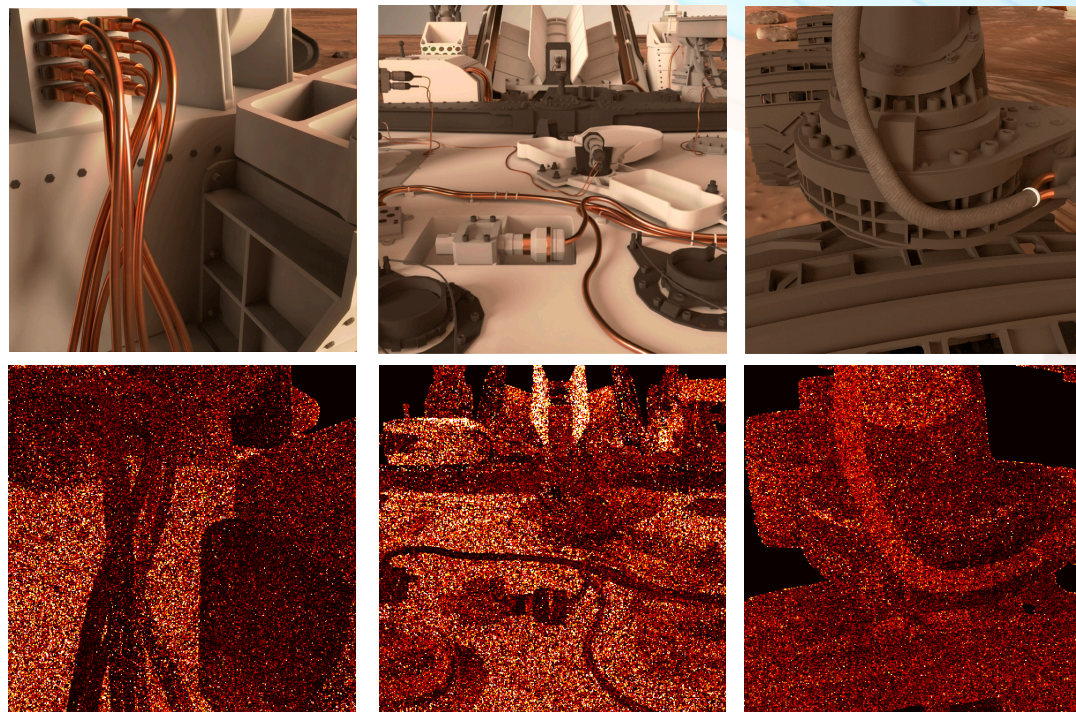
[Miller et al. 2024]

# Monte Carlo PDE solvers

Mars curiosity rover



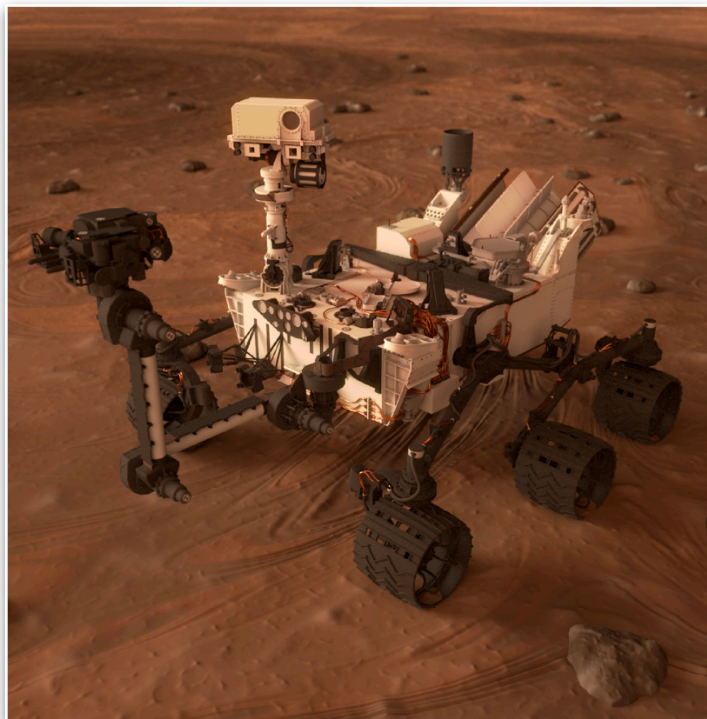
close up with thermal simulation results



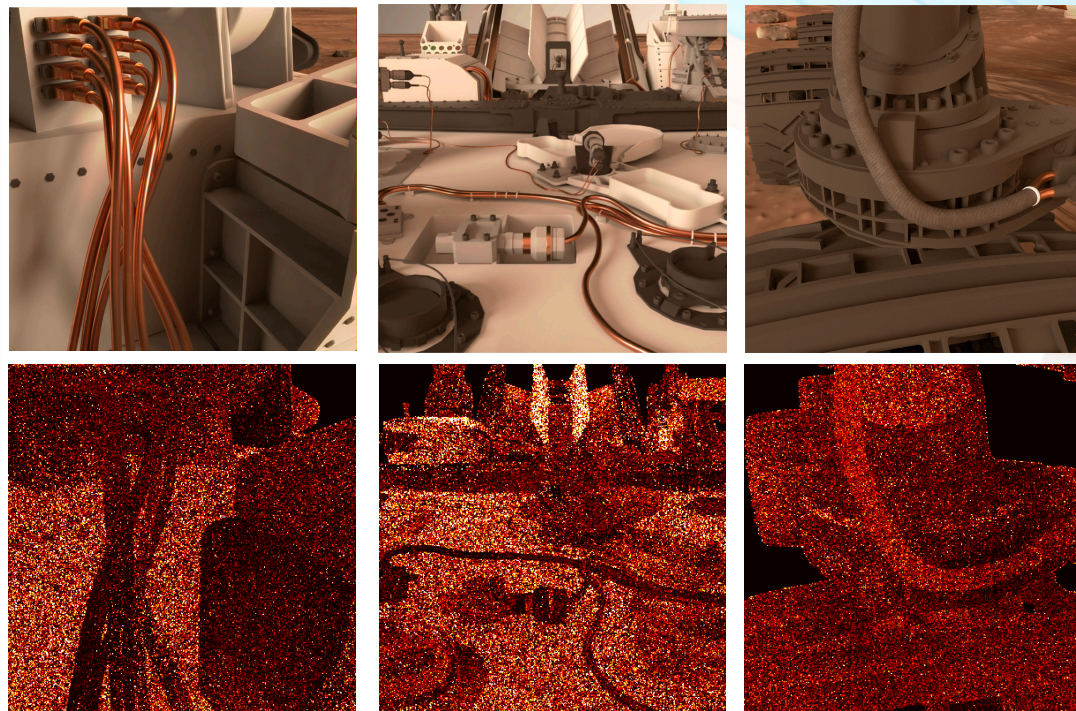
[Miller et al. 2024]

# Monte Carlo PDE solvers

Mars curiosity rover



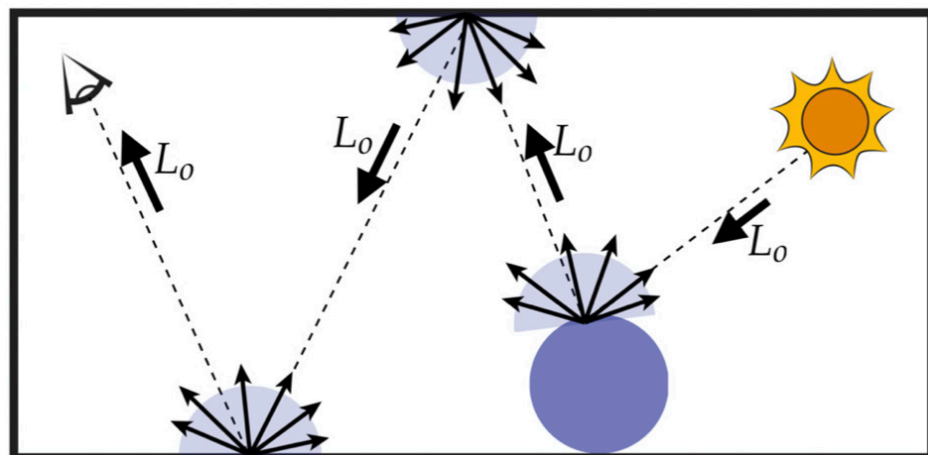
close up with thermal simulation results



[Miller et al. 2024]

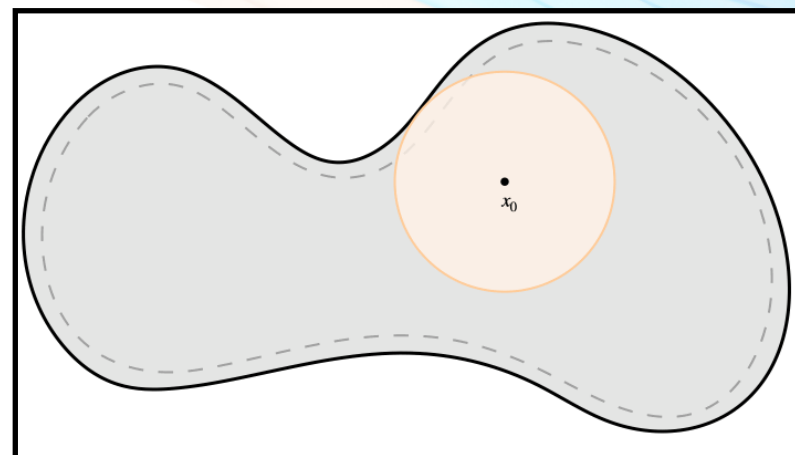
# Monte Carlo PDE solvers

rendering



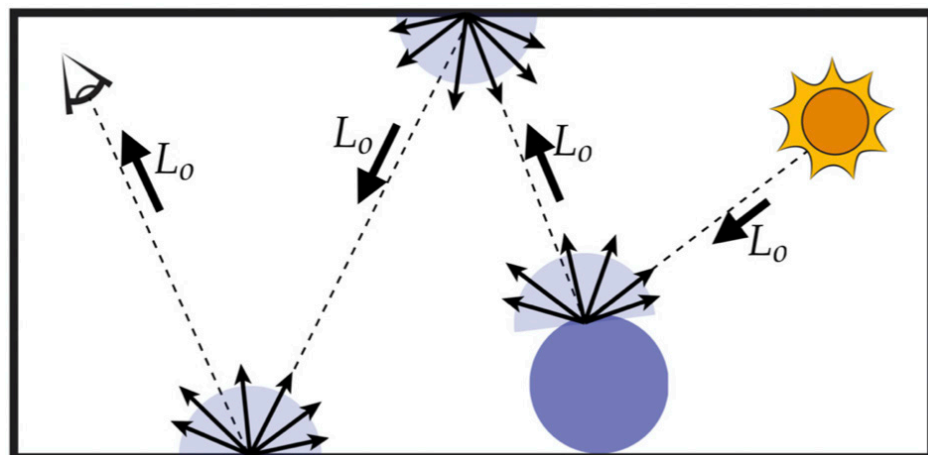
walk on spheres

[Muller 1956, Sawhney and Crane 2020]



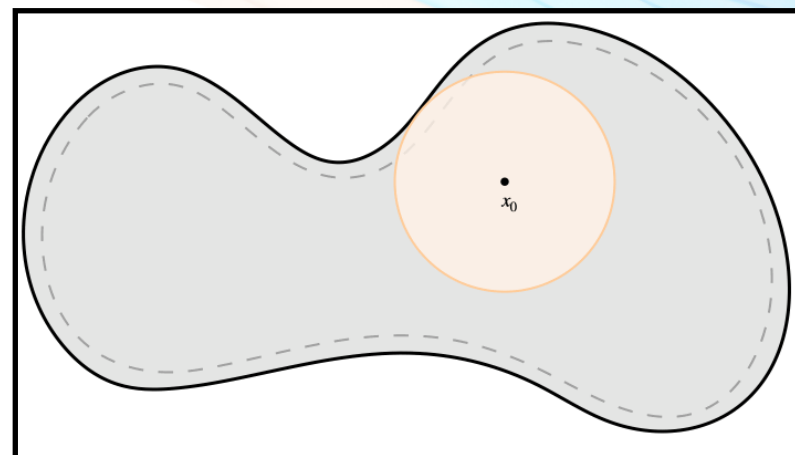
# Monte Carlo PDE solvers

rendering



walk on spheres

[Muller 1956, Sawhney and Crane 2020]



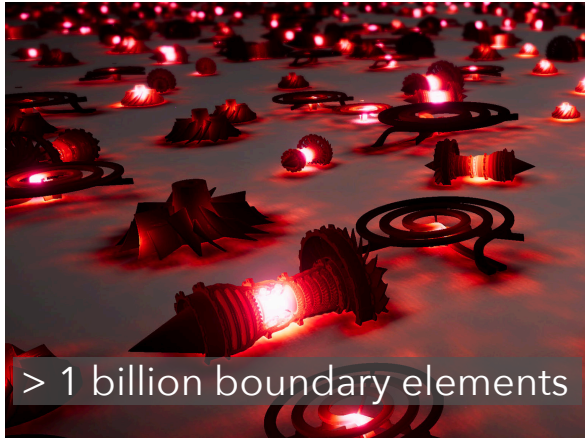


# benefits of Monte Carlo PDE solvers



# benefits of Monte Carlo PDE solvers

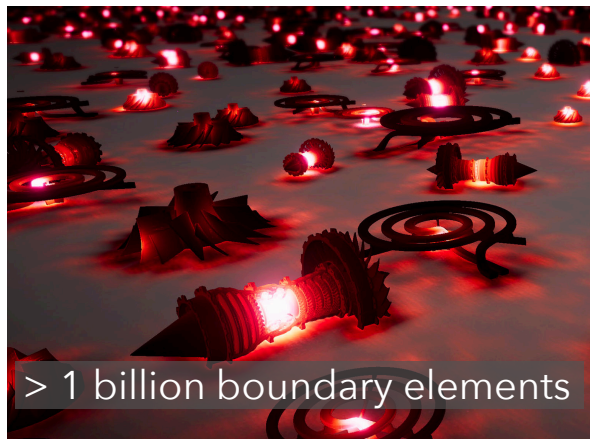
geometric scalability



[Sawhney et al. 2022]

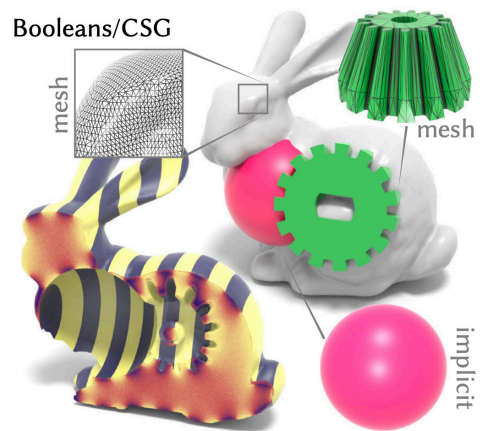
# benefits of Monte Carlo PDE solvers

geometric scalability



[Sawhney et al. 2022]

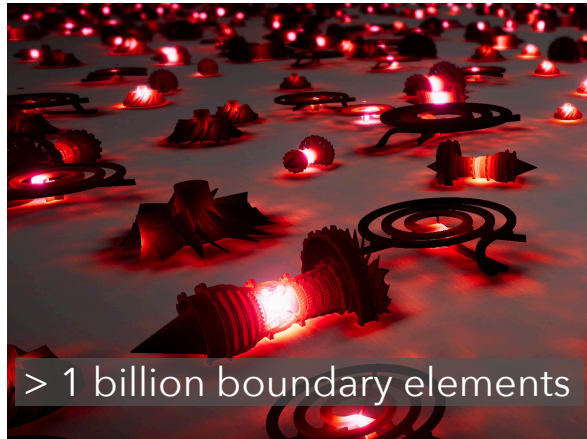
flexible representations



[Sawhney and Crane 2020]

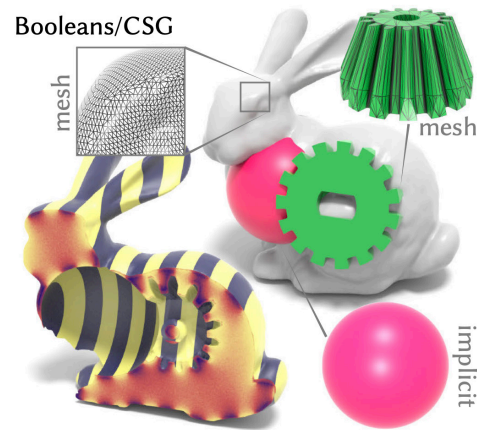
# benefits of Monte Carlo PDE solvers

geometric scalability



[Sawhney et al. 2022]

flexible representations



[Sawhney and Crane 2020]

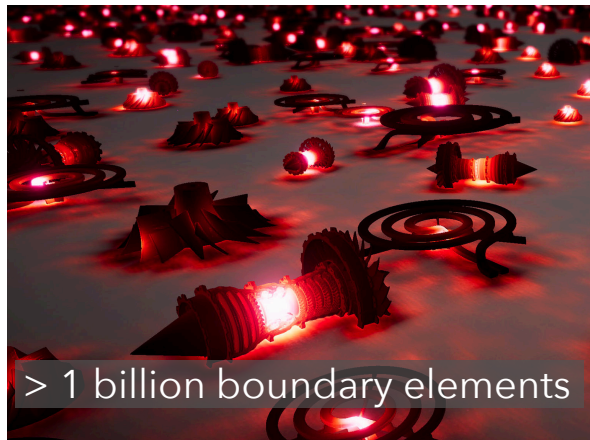
fast noisy previews



[Sawhney et al. 2023]

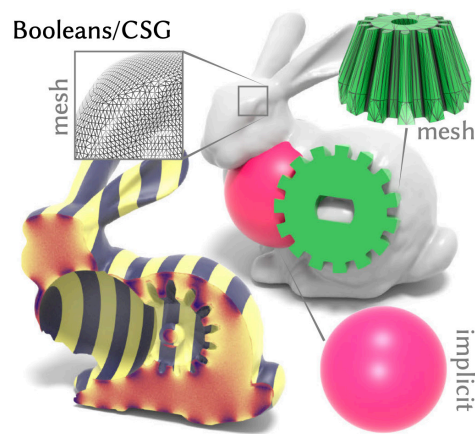
# benefits of Monte Carlo PDE solvers

geometric scalability



[Sawhney et al. 2022]

flexible representations



[Sawhney and Crane 2020]

fast noisy previews



[Sawhney et al. 2023]

parallelizability



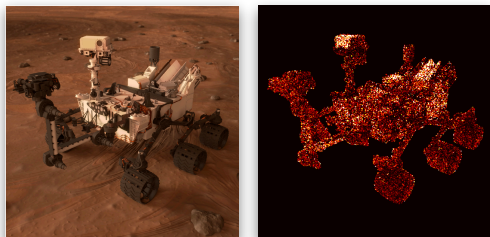
# increasing capabilities of Monte Carlo solvers



# increasing capabilities of Monte Carlo solvers

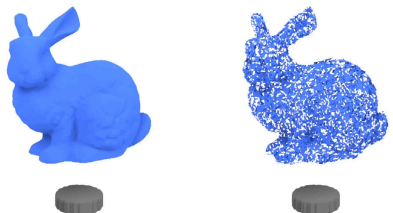
expanded generality

boundary conditions



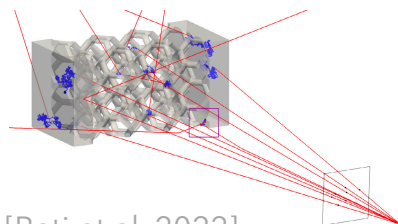
[Miller et al. 2024]

physical phenomena



[Rioux-Lavoie et al. 2022]

coupled physics

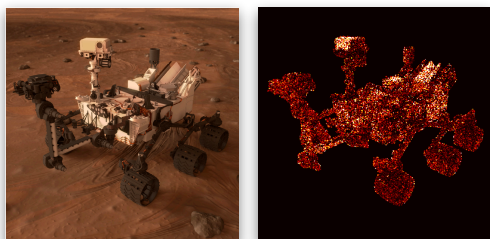


[Bati et al. 2022]

# increasing capabilities of Monte Carlo solvers

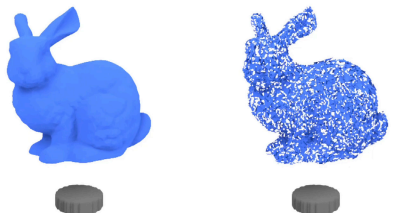
expanded generality

boundary conditions



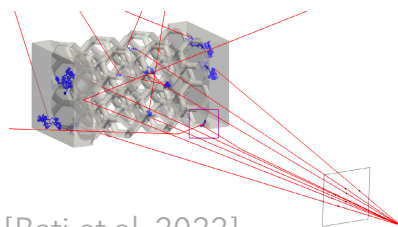
[Miller et al. 2024]

physical phenomena



[Rioux-Lavoie et al. 2022]

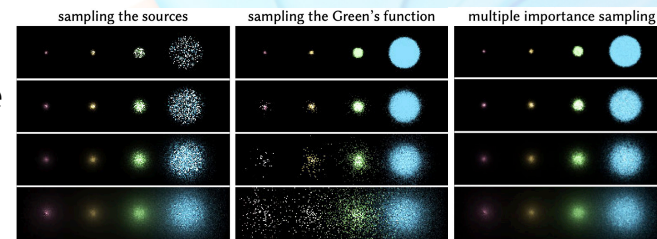
coupled physics



[Bati et al. 2022]

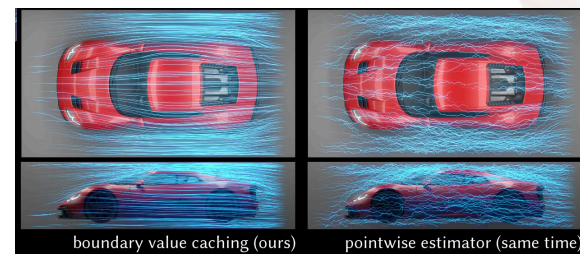
improved performance

importance sampling



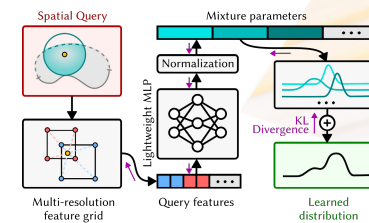
[Sawhney and Crane 2020]

caching schemes



[Miller et al. 2023]

path guiding



[Huang et al. 2024]

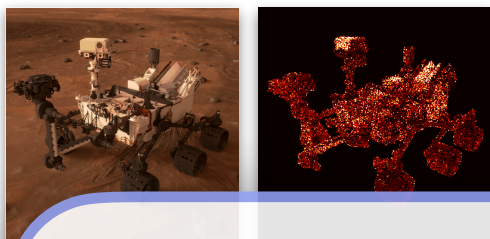


# increasing capabilities of Monte Carlo solvers

expanded generality

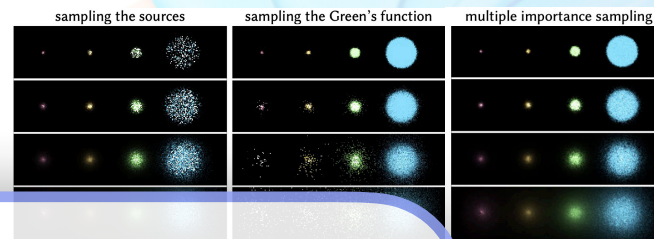
improved performance

boundary conditions



[Miller et al. 2024]

importance sampling



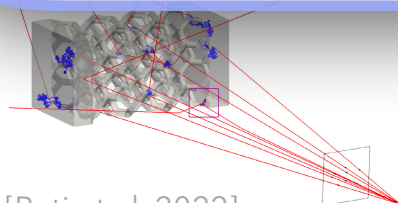
[Sawhney and Crane 2020]

physical phenomena

**Limited work on differentiable Monte Carlo PDE solvers!**

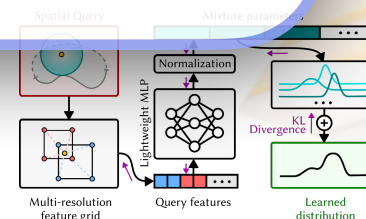
[Rioux-Lavoie et al. 2022]

coupled physics



[Bati et al. 2022]

path guiding



[Huang et al. 2024]

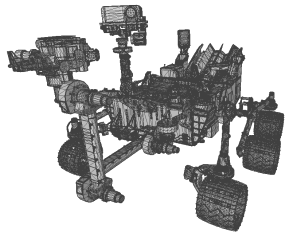
# forward PDE solver

physical model parameters  $[\pi_g, \pi_f, \pi_m]$

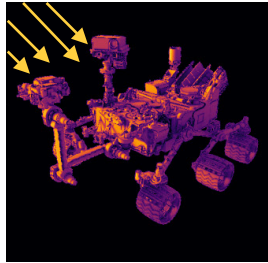
# forward PDE solver

simulation result

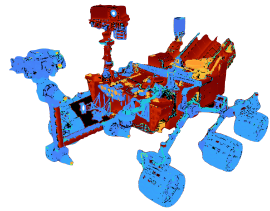
physical model parameters  $[\pi_g, \pi_f, \pi_m]$



geometry  $\pi_g$



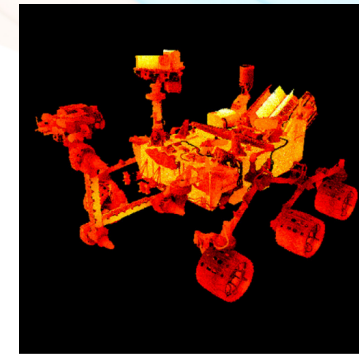
boundary conditions  $\pi_f$



materials  $\pi_m$

$$u(x, [\pi_g, \pi_f, \pi_m])$$

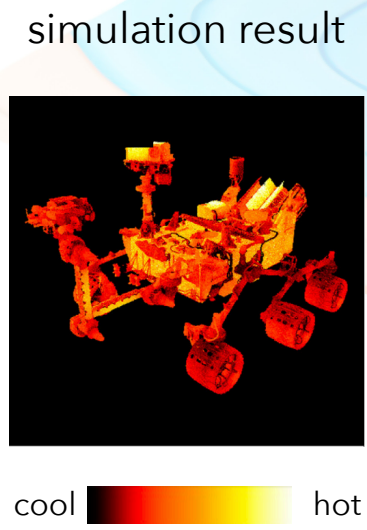
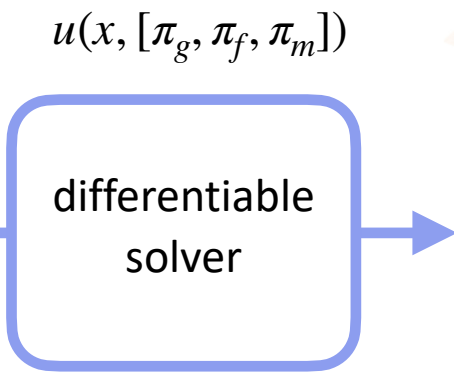
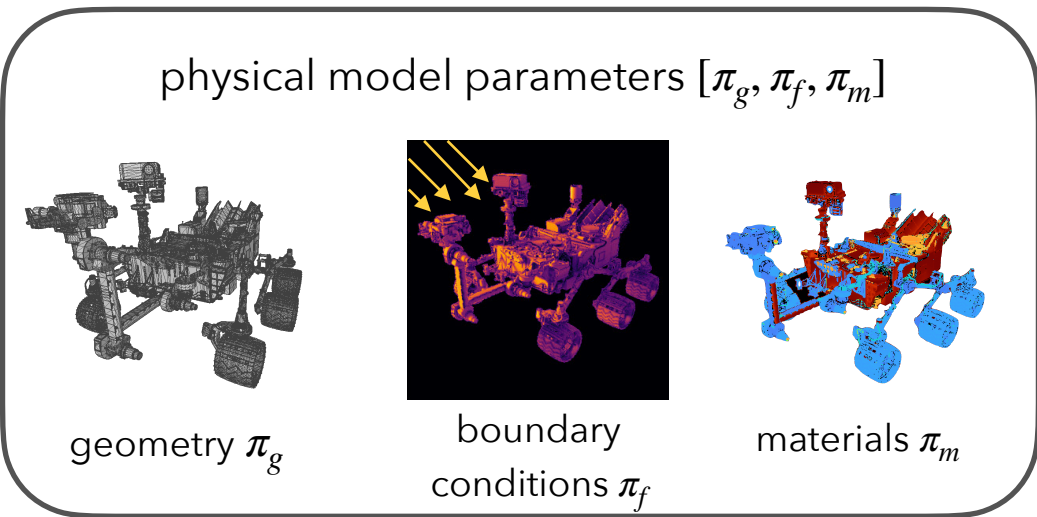
forward solver



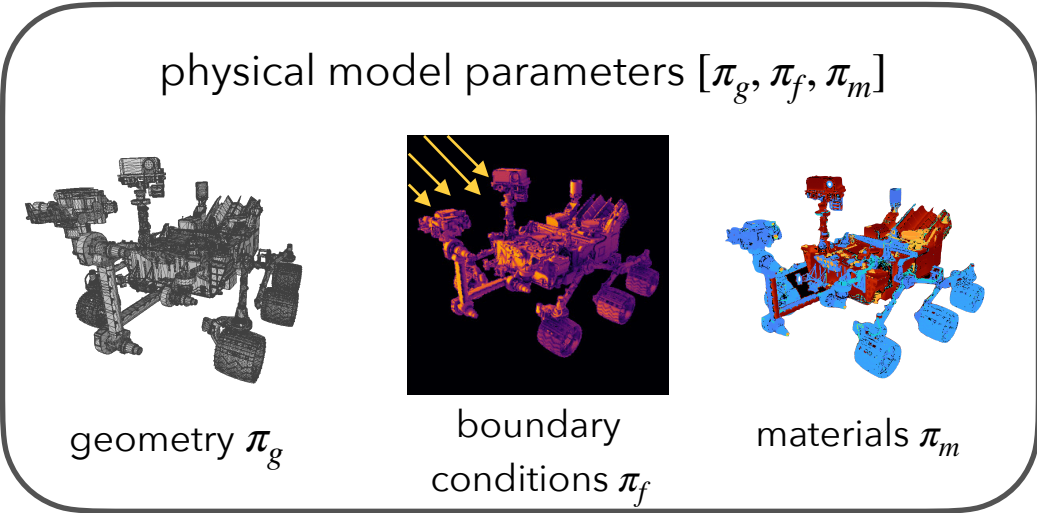
cool  hot



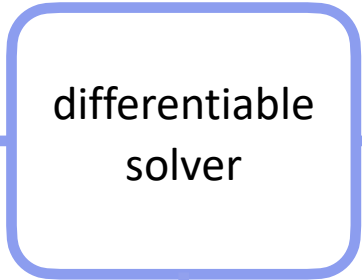
# differentiable PDE solver



# differentiable PDE solver

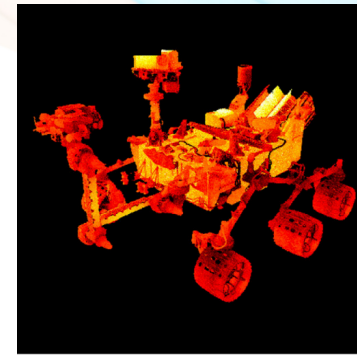


$$u(x, [\pi_g, \pi_f, \pi_m])$$



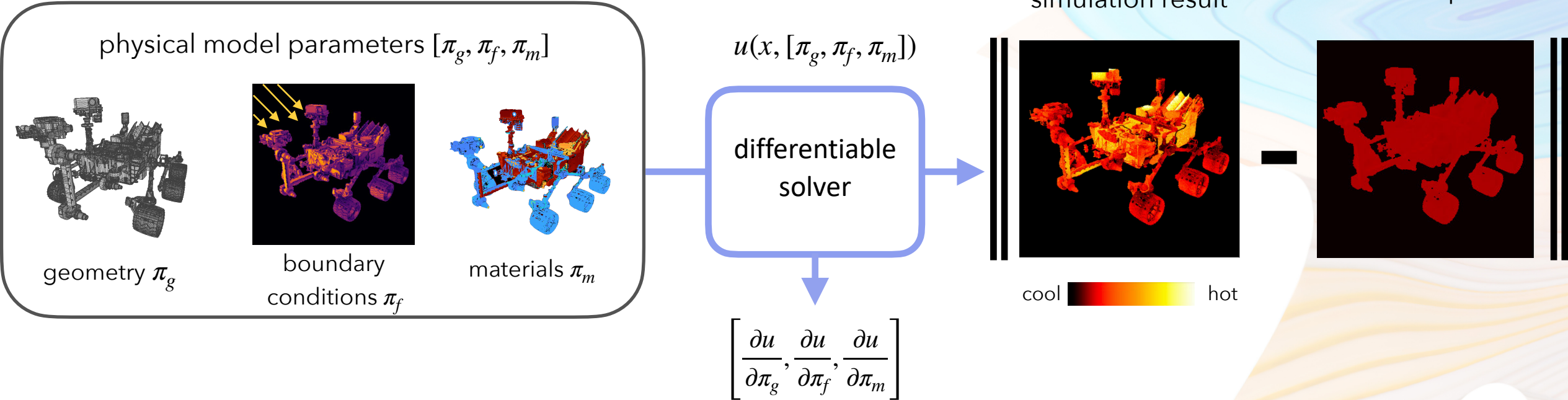
$$\left[ \frac{\partial u}{\partial \pi_g}, \frac{\partial u}{\partial \pi_f}, \frac{\partial u}{\partial \pi_m} \right]$$

simulation result

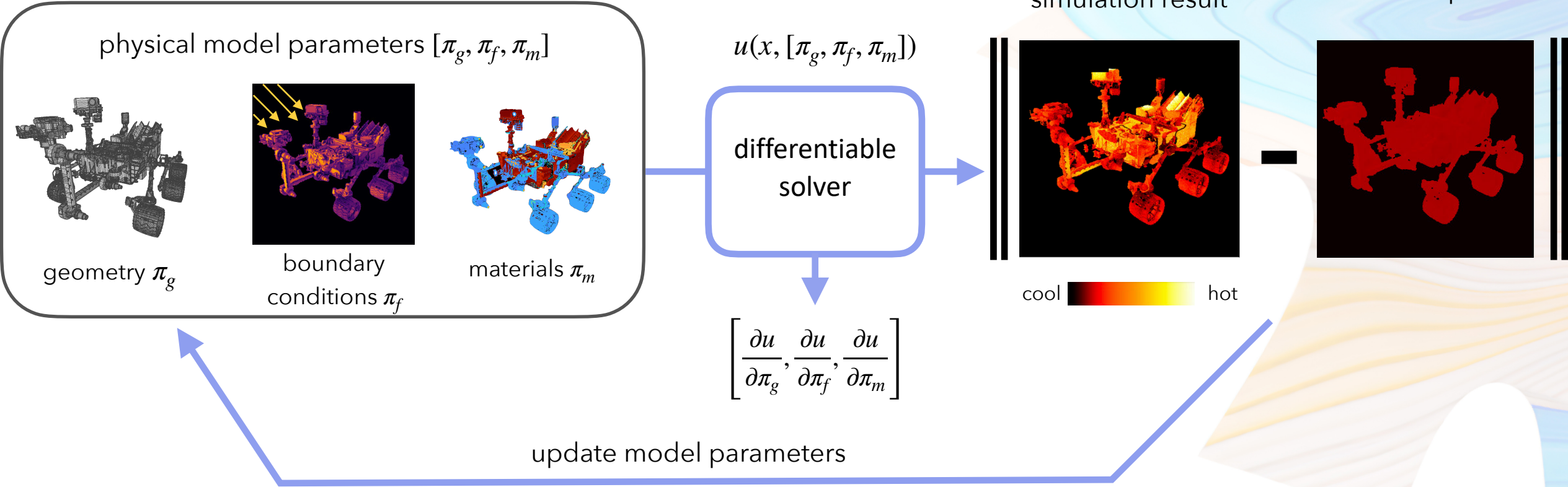


cool  hot

# differentiable PDE solver



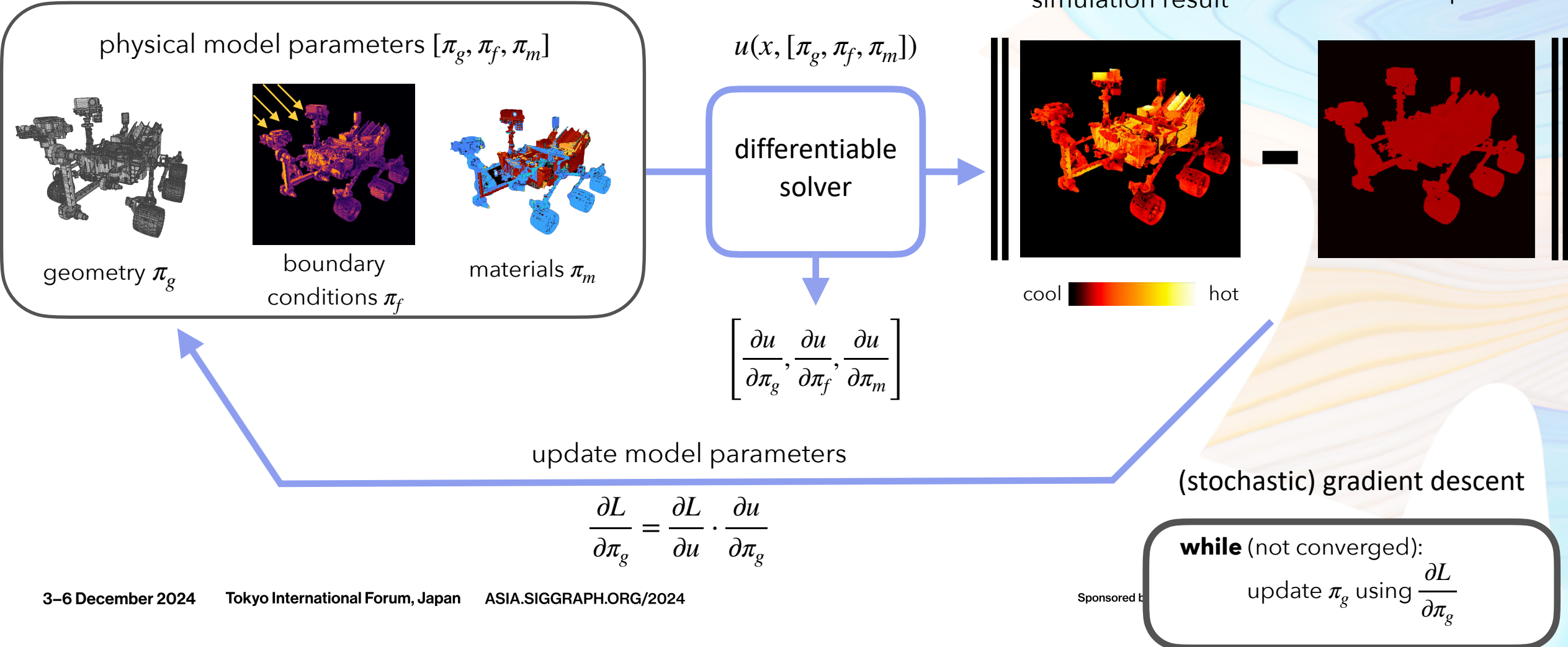
# differentiable PDE solver



$$\left[ \frac{\partial u}{\partial \pi_g}, \frac{\partial u}{\partial \pi_f}, \frac{\partial u}{\partial \pi_m} \right]$$

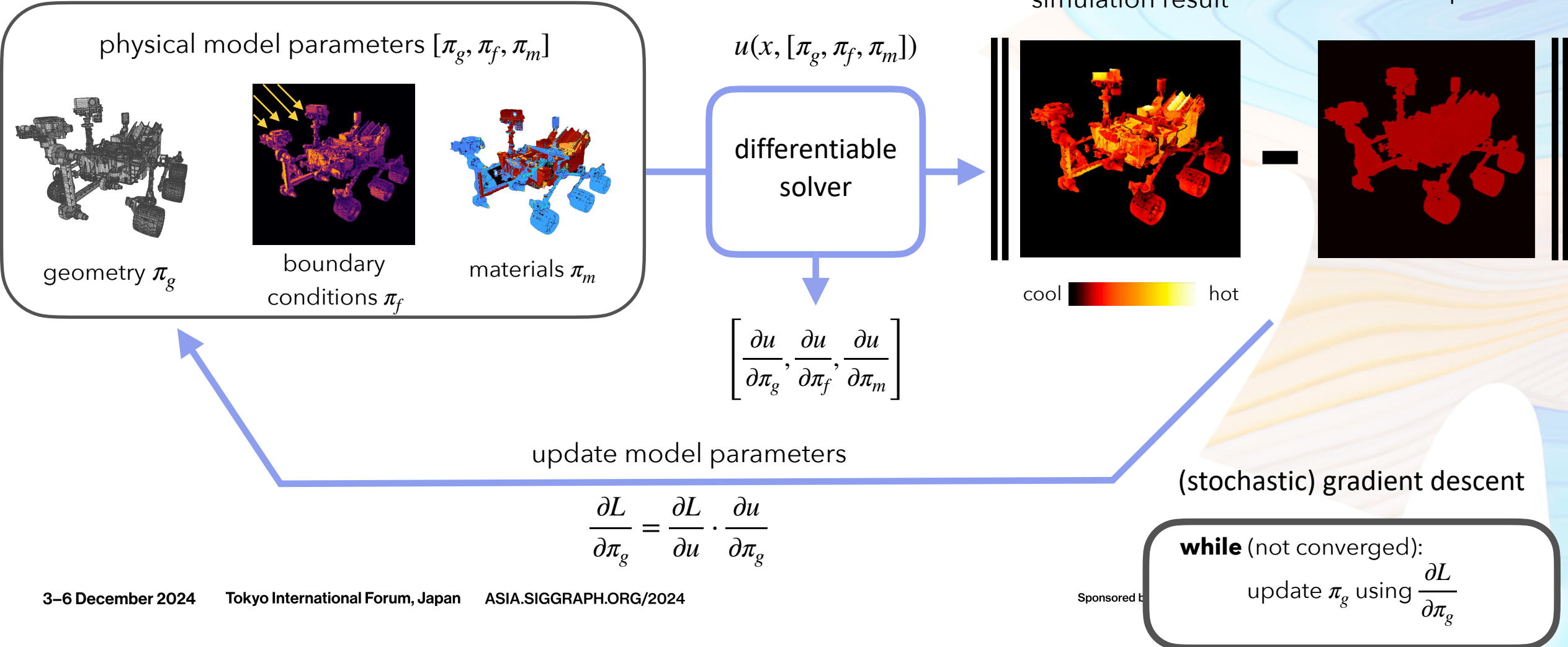
$$\frac{\partial L}{\partial \pi_g} = \frac{\partial L}{\partial u} \cdot \frac{\partial u}{\partial \pi_g}$$

# differentiable PDE solver

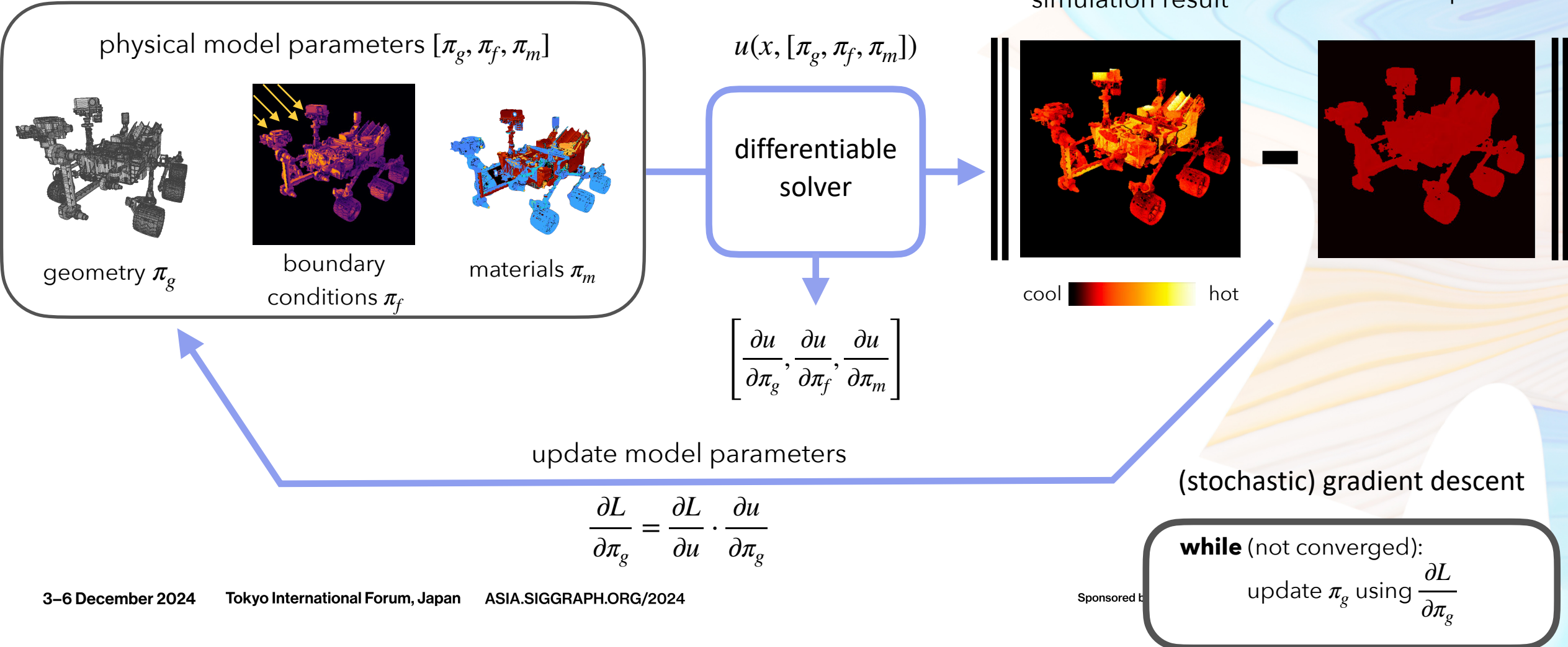




# differentiable PDE solver

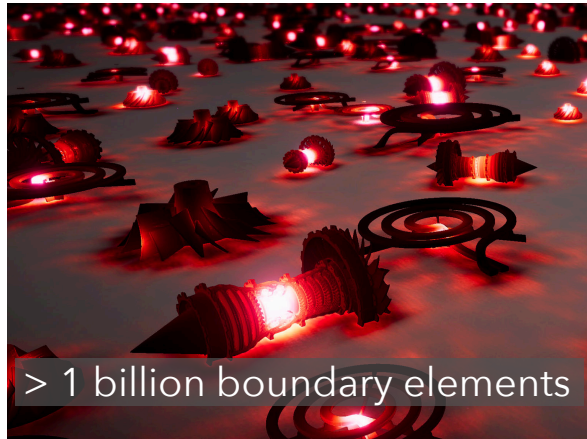


# differentiable PDE solver



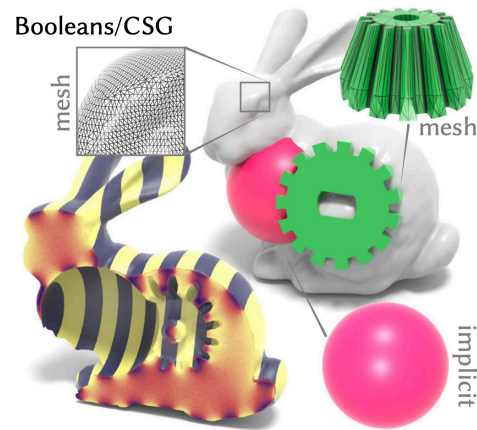
# Monte Carlo well suited for differentiability

geometric scalability



[Sawhney et al. 2022]

flexible representations



[Sawhney and Crane 2020]

fast noisy previews



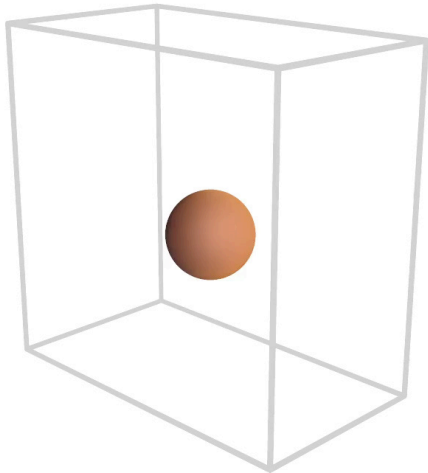
[Sawhney et al. 2023]

parallelizability

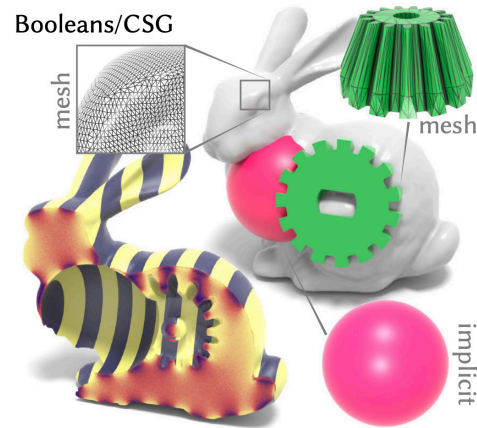


# Monte Carlo well suited for differentiability

geometric scalability



flexible representations



[Sawhney and Crane 2020]

fast noisy previews



[Sawhney et al. 2023]

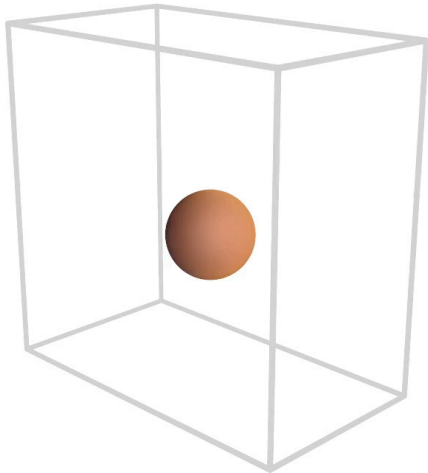
parallelizability



→ avoid repeatedly creating volume mesh

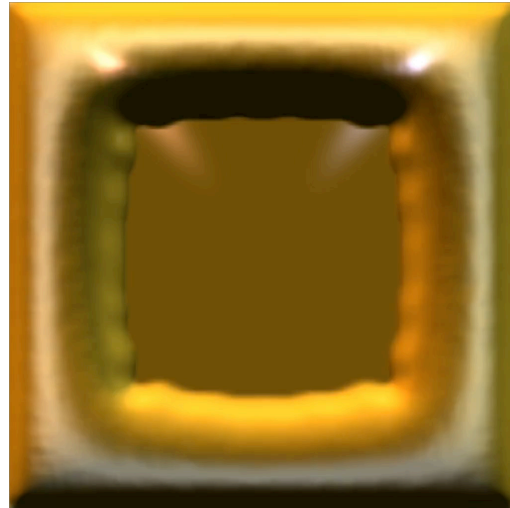
# Monte Carlo well suited for differentiability

geometric scalability



→ avoid repeatedly creating volume mesh

flexible representations



→ implicit surfaces easily change topology

fast noisy previews



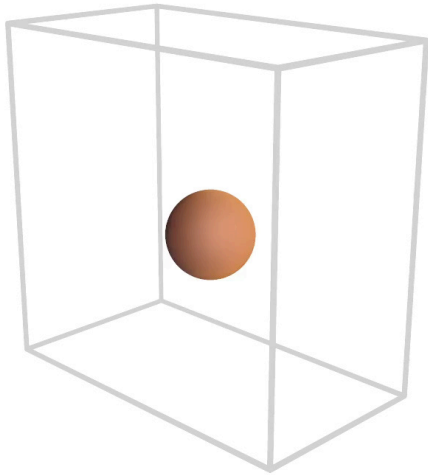
[Sawhney et al. 2023]

parallelizability



# Monte Carlo well suited for differentiability

geometric scalability



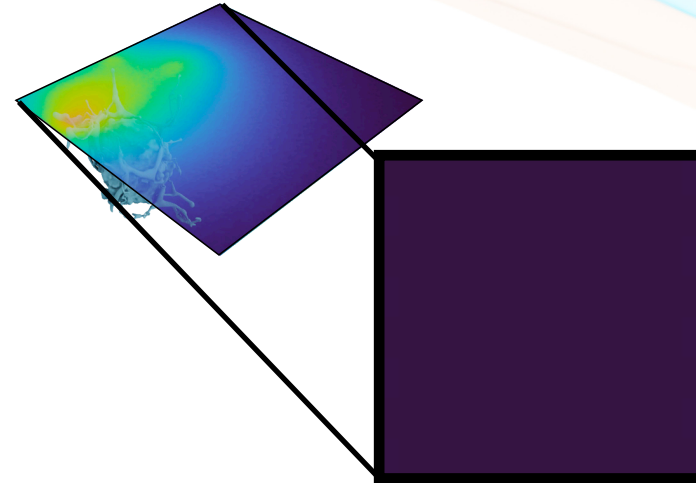
→ avoid repeatedly creating volume mesh

flexible representations



→ implicit surfaces easily change topology

fast noisy previews



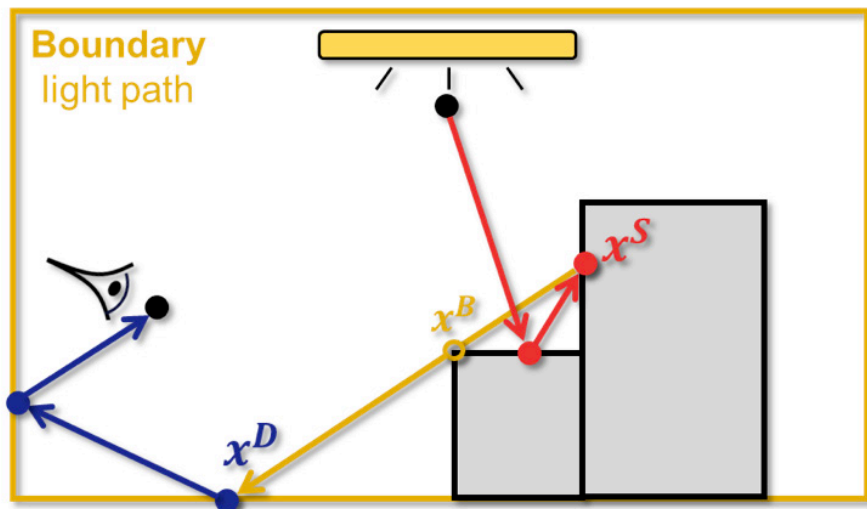
→ use noisy gradient estimates early in optimization (stochastic gradients)

parallelizability

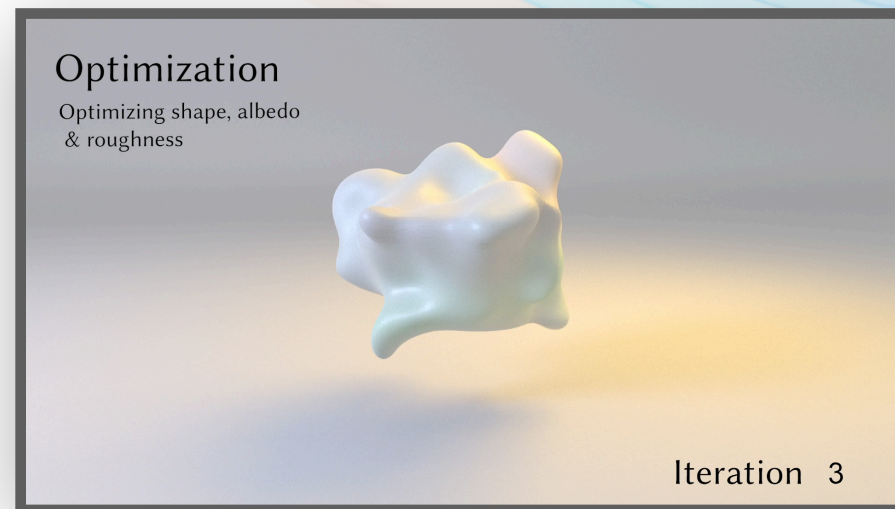


# similar to differentiable rendering

differentiable rendering



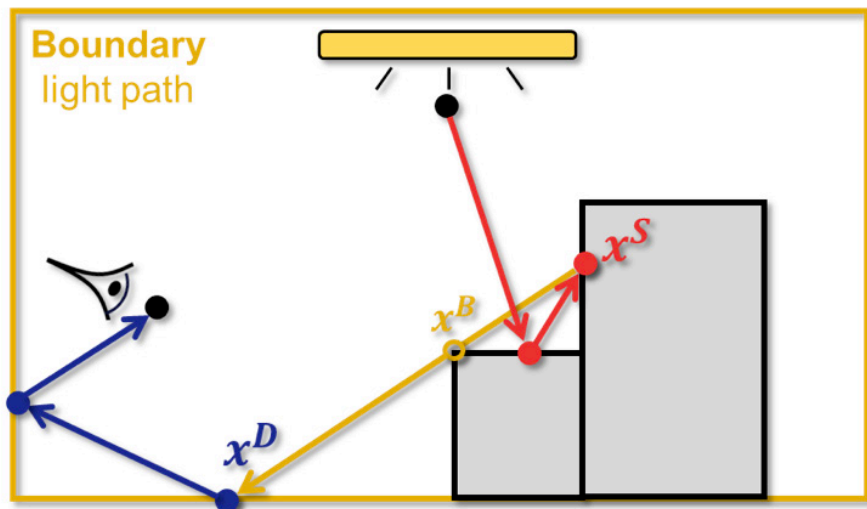
[Zhang et al. 2020]



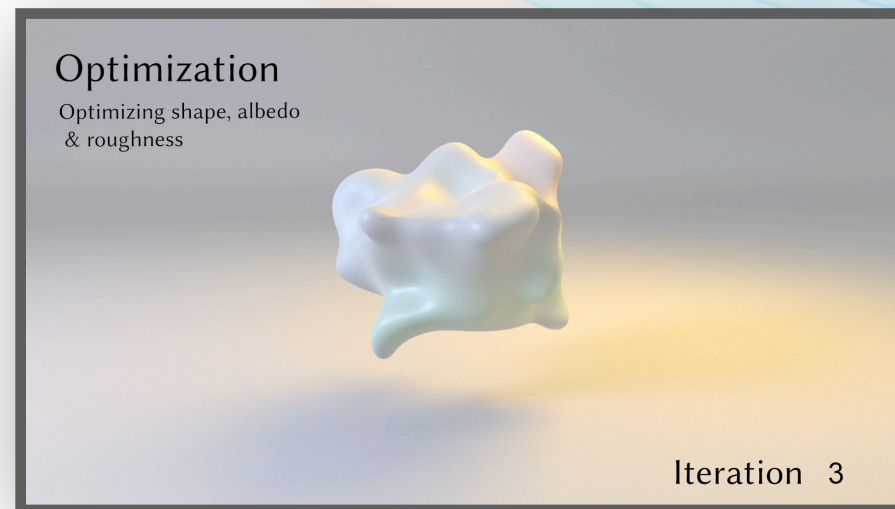
[Vicini et al. 2023]

# similar to differentiable rendering

differentiable rendering



[Zhang et al. 2020]

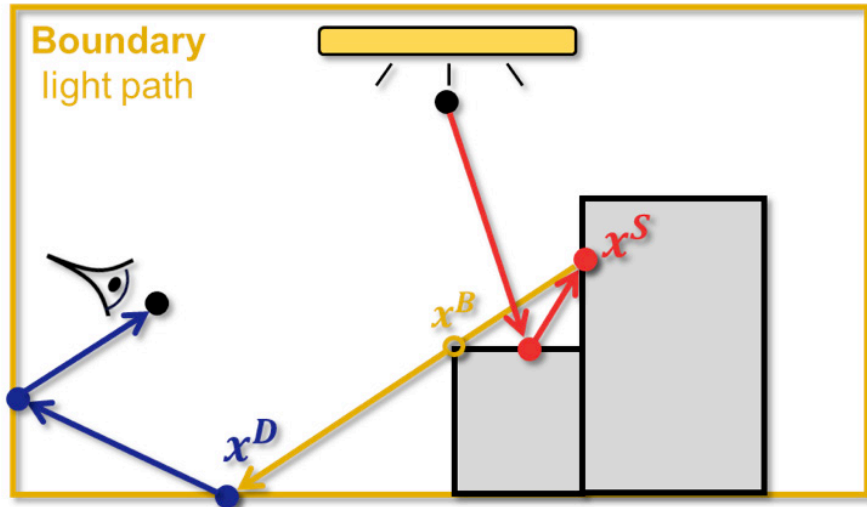


[Vicini et al. 2023]



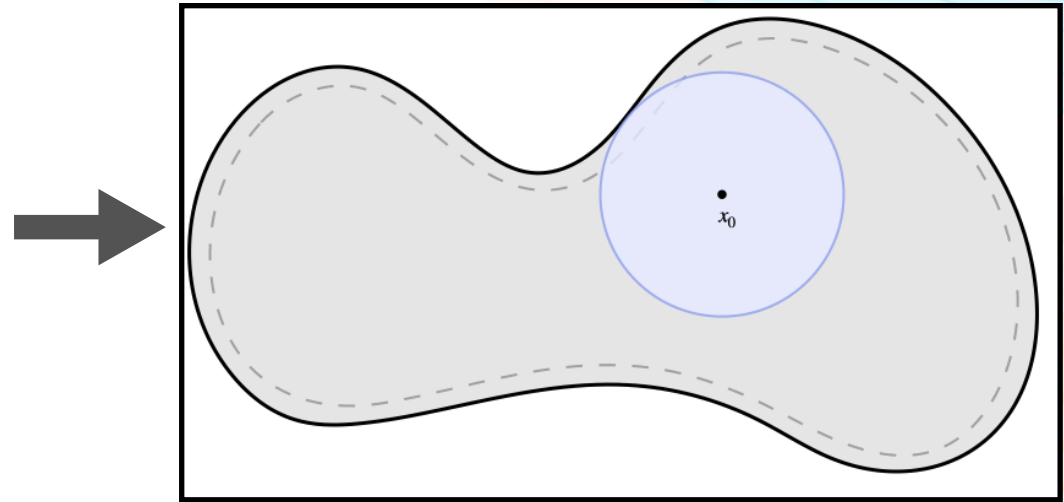
# similar to differentiable rendering

differentiable rendering



[Zhang et al. 2020]

differential walk on spheres



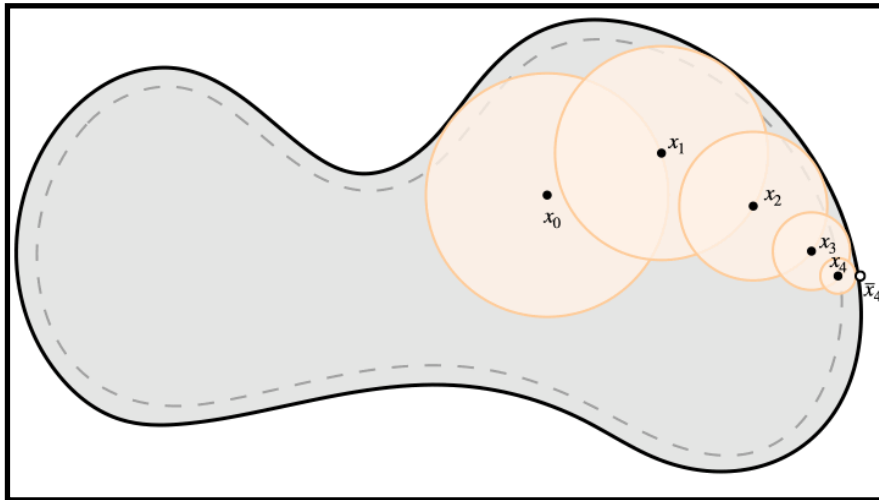
# differential walk on spheres

primal

PDE

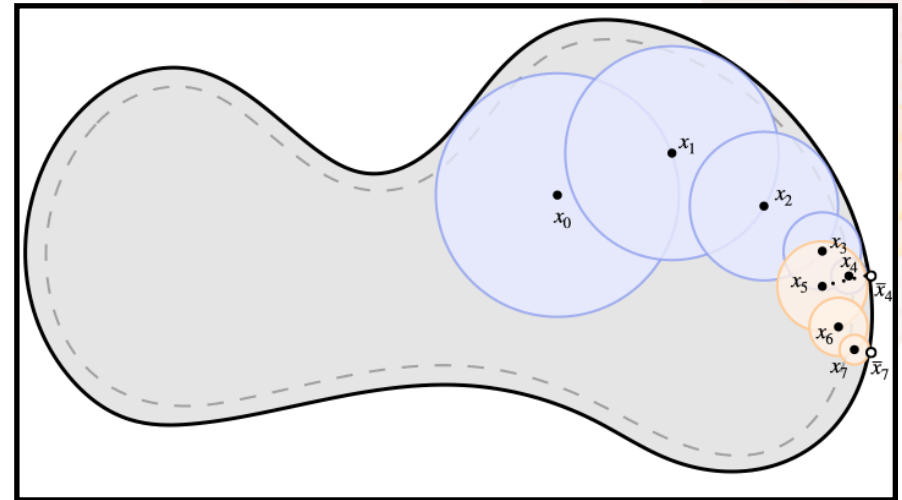
$$\begin{aligned} \Delta u &= 0 && \text{on } \Omega \\ u &= g && \text{on } \partial\Omega \end{aligned}$$

algorithm



differential

$$\begin{aligned} \Delta \dot{u} &= 0 && \text{on } \Omega \\ \dot{u} &= V_n \left( \frac{\partial g}{\partial n} - \frac{\partial u}{\partial n} \right) && \text{on } \partial\Omega \end{aligned}$$



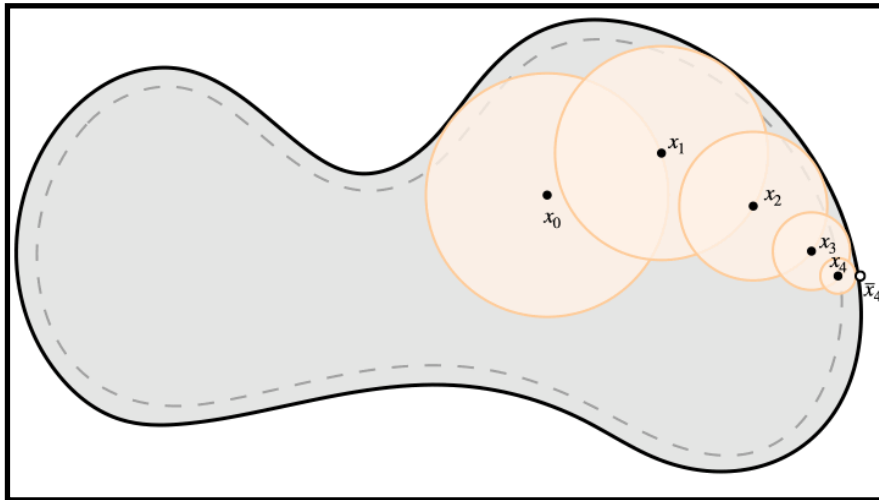
# differential walk on spheres

primal

PDE

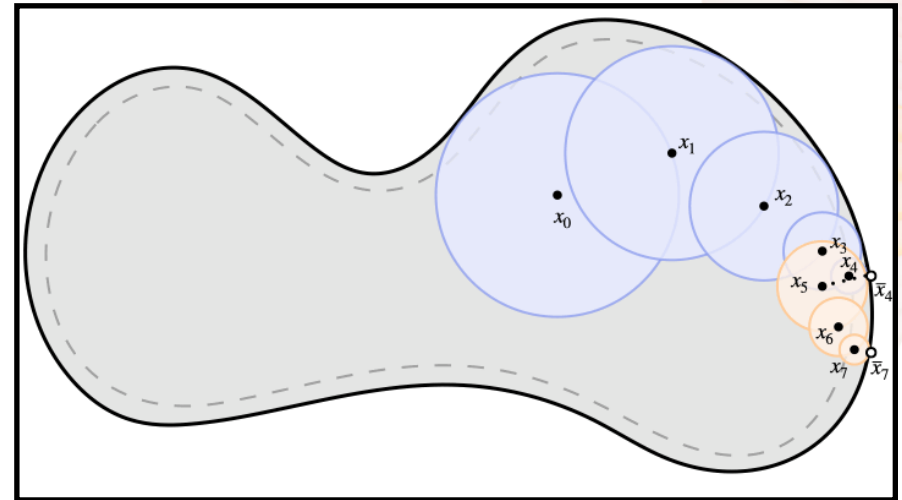
$$\begin{aligned} \Delta u &= 0 && \text{on } \Omega \\ u &= g && \text{on } \partial\Omega \end{aligned}$$

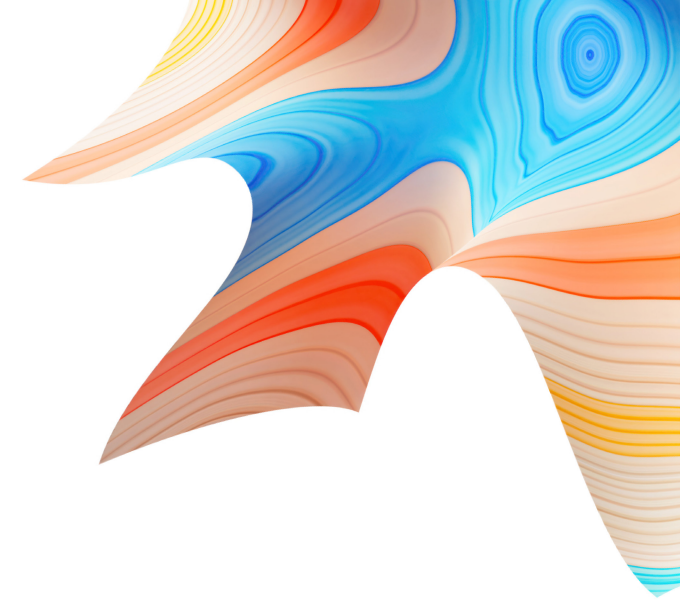
algorithm



differential

$$\begin{aligned} \Delta \dot{u} &= 0 && \text{on } \Omega \\ \dot{u} &= V_n \left( \frac{\partial g}{\partial n} - \frac{\partial u}{\partial n} \right) && \text{on } \partial\Omega \end{aligned}$$





# review of walk on spheres



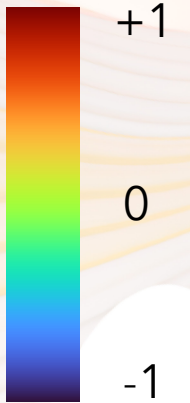
# Laplace PDE with Dirichlet boundary cond.

$$\Delta u = 0 \quad \text{on } \Omega$$

$$u = g \quad \text{on } \partial\Omega$$

$g(x)$

$\partial\Omega$

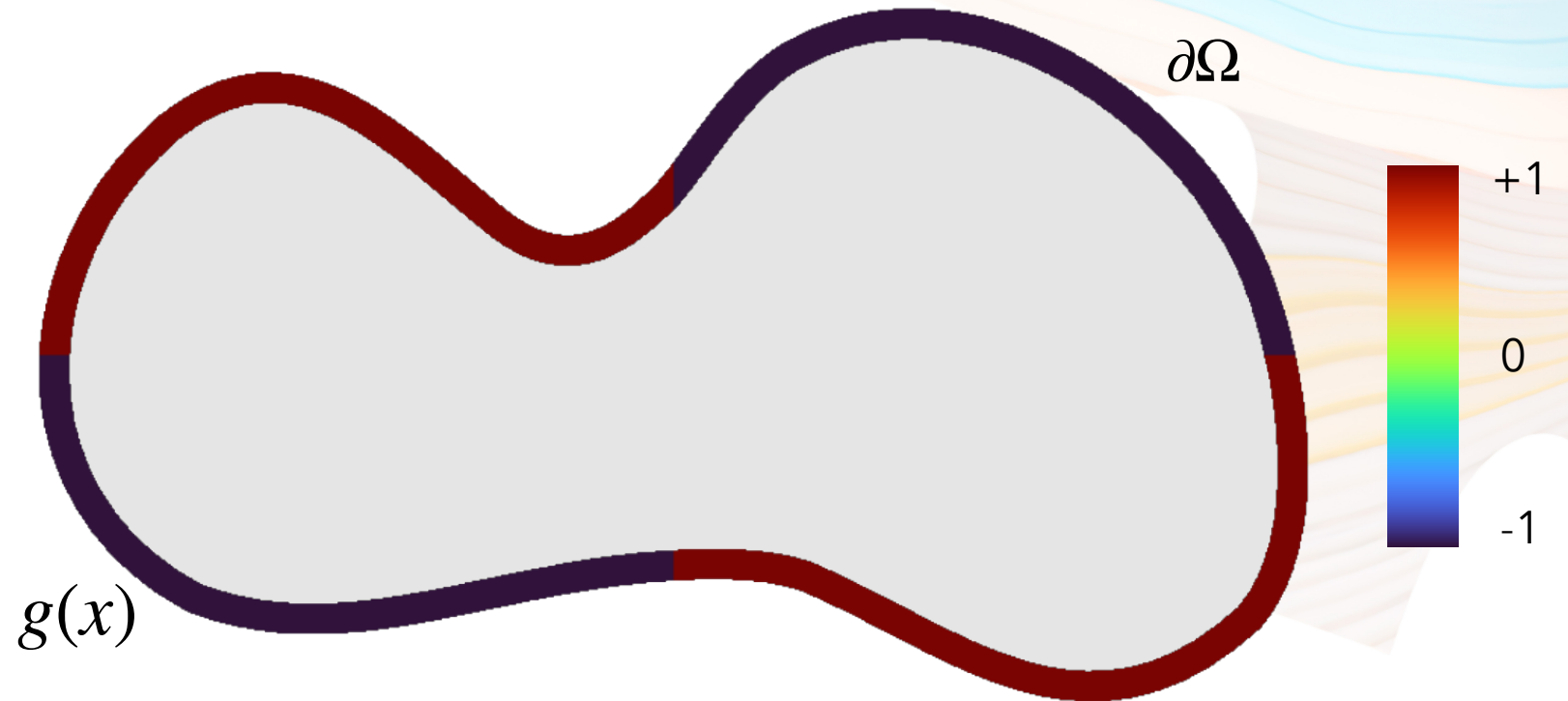


# Laplace PDE with Dirichlet boundary cond.

$$\Delta u = 0 \quad \text{on } \Omega$$

$$u = g \quad \text{on } \partial\Omega$$

**given:** values on the boundary of a region  $\Omega$



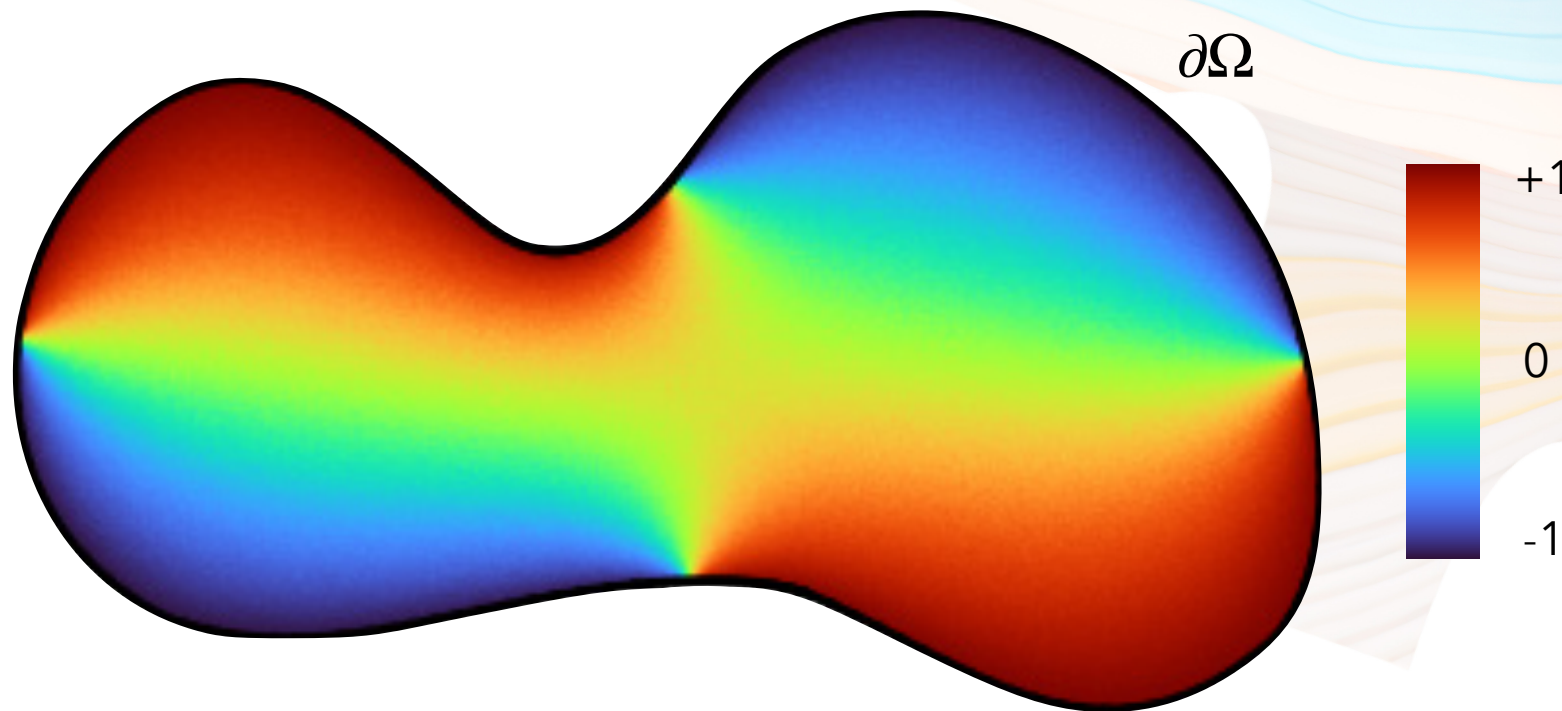
# Laplace PDE with Dirichlet boundary cond.

$$\Delta u = 0 \quad \text{on } \Omega$$

$$u = g \quad \text{on } \partial\Omega$$

**given:** values on the boundary of a region  $\Omega$

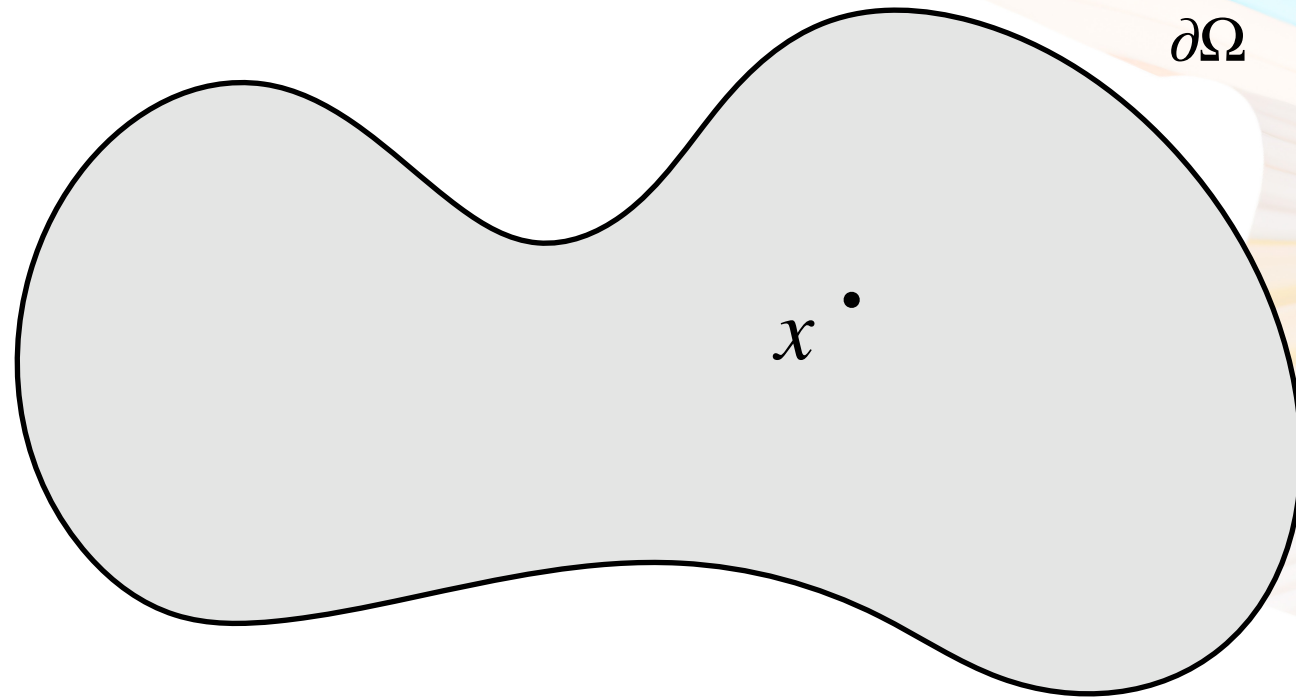
**find:** smooth interpolation into interior



# mean value integral

$$u(x) = \frac{1}{|\partial B(x)|} \int_{\partial B(x)} u(y) dy$$

since  $\Delta u = 0$

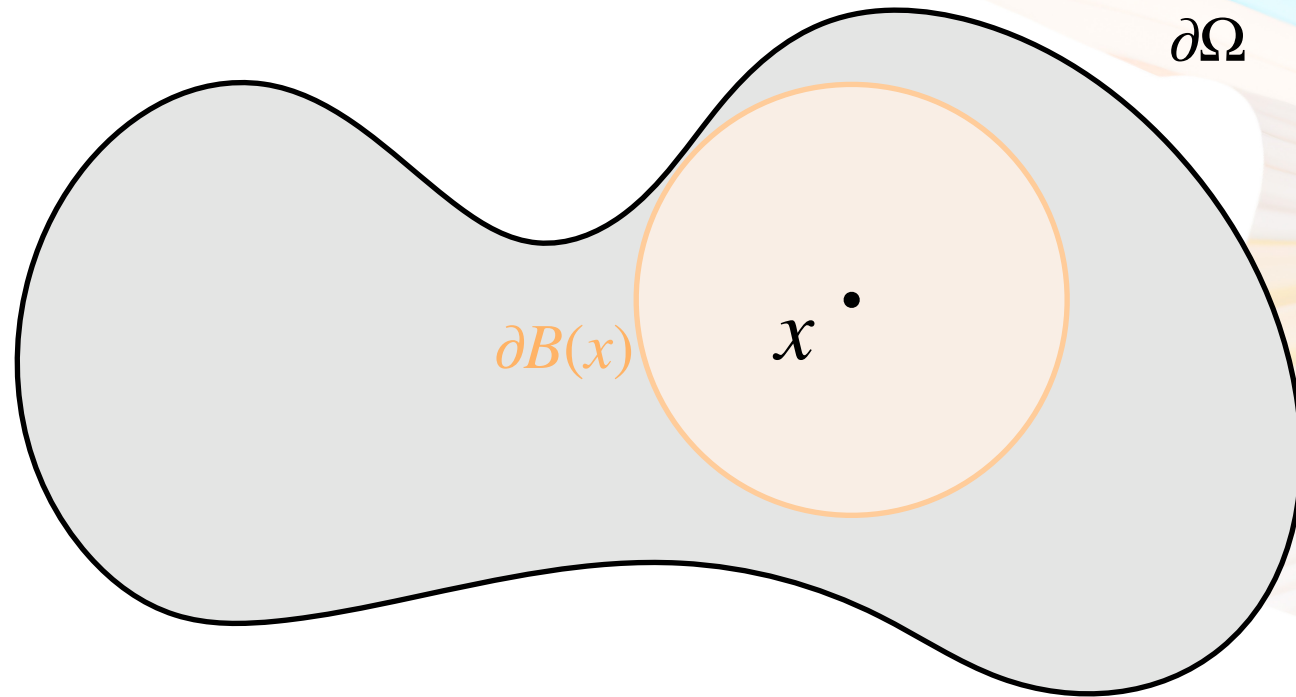




# mean value integral

$$u(x) = \frac{1}{|\partial B(x)|} \int_{\partial B(x)} u(y) dy$$

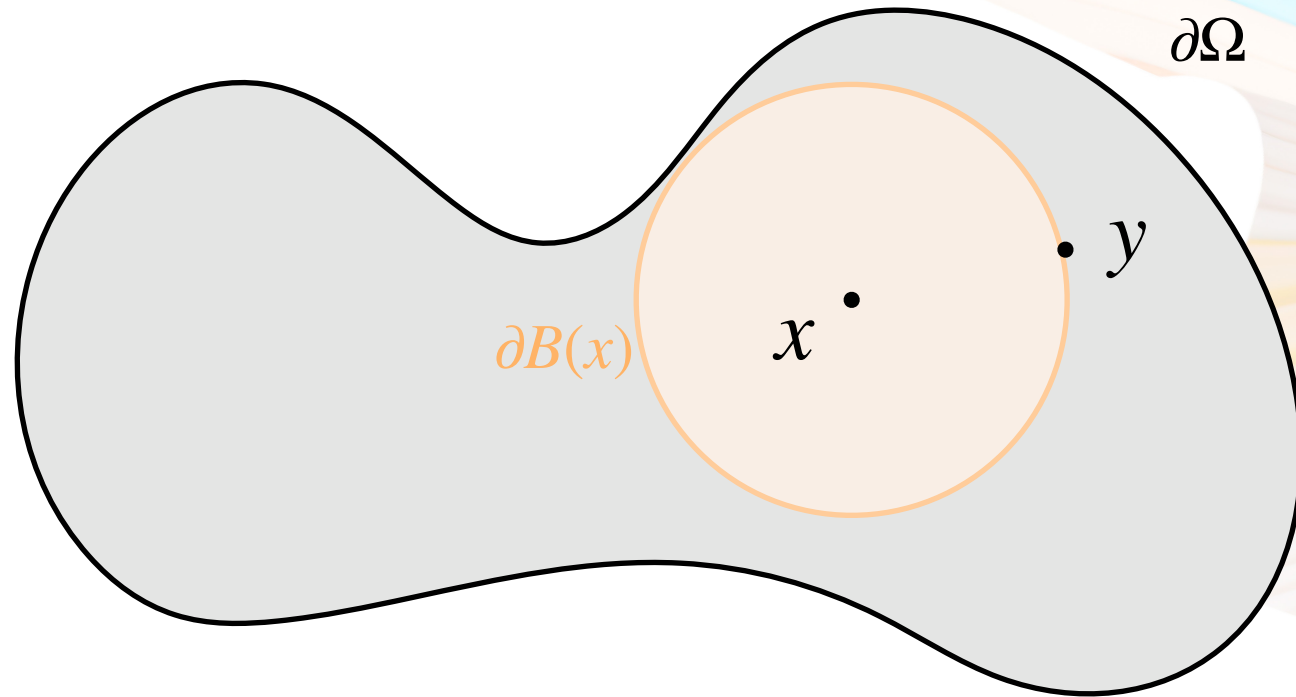
since  $\Delta u = 0$



# Monte Carlo estimator

$$\hat{u}(x) = \begin{cases} \hat{g}(y) & \text{if } y \in \partial\Omega \\ \hat{u}(y) & \text{otherwise} \end{cases}$$

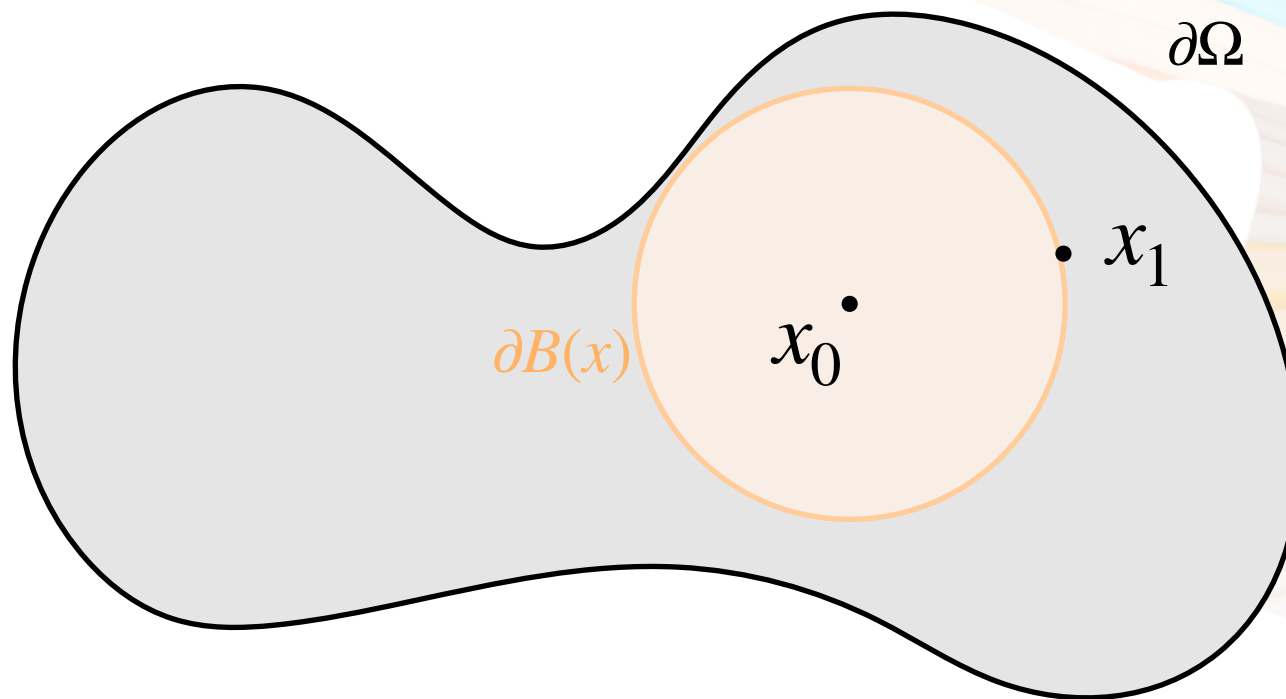
$$y \sim \mathcal{U} [\partial B(x)]$$



# walk on spheres [Muller 1956, Sawhney and Crane 2020]

$$\hat{u}(x_i) = \begin{cases} \hat{g}(x_{i+1}) & \text{if } x_{i+1} \in \partial\Omega_\epsilon \\ \hat{u}(x_{i+1}) & \text{otherwise} \end{cases}$$

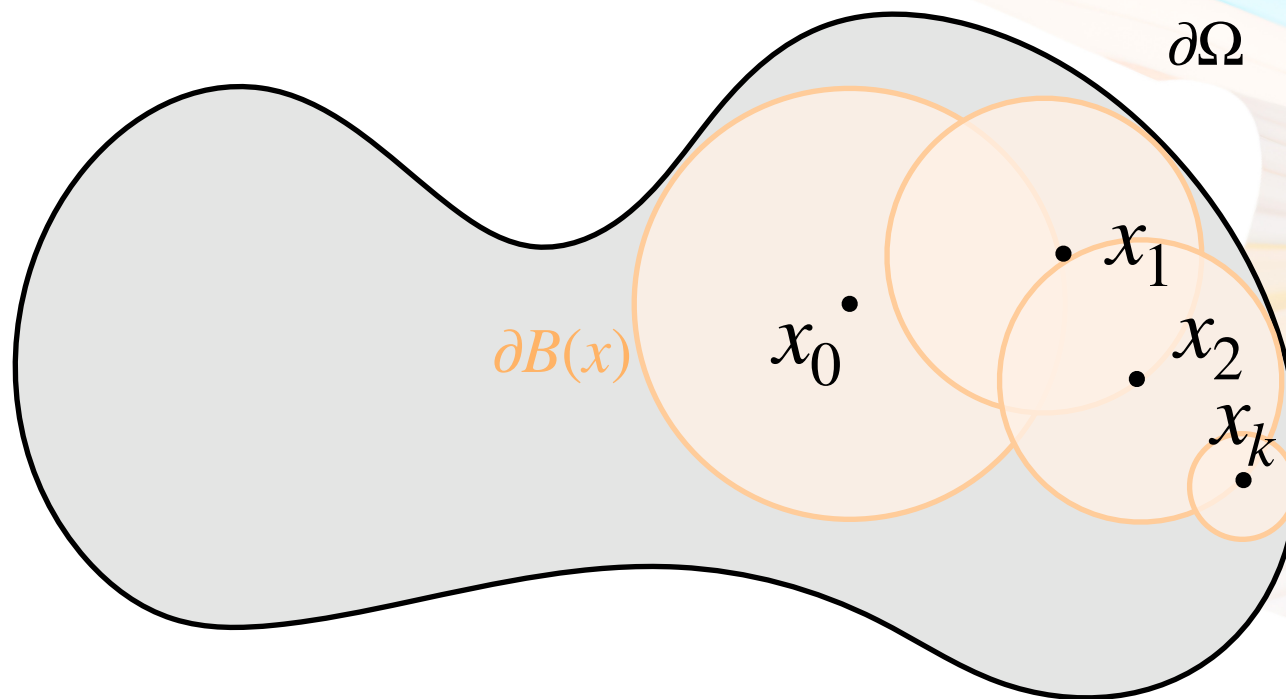
$$x_{i+1} \sim \mathcal{U} [\partial B(x_i)]$$



# walk on spheres [Muller 1956, Sawhney and Crane 2020]

$$\hat{u}(x_i) = \begin{cases} \hat{g}(x_{i+1}) & \text{if } x_{i+1} \in \partial\Omega_\epsilon \\ \hat{u}(x_{i+1}) & \text{otherwise} \end{cases}$$

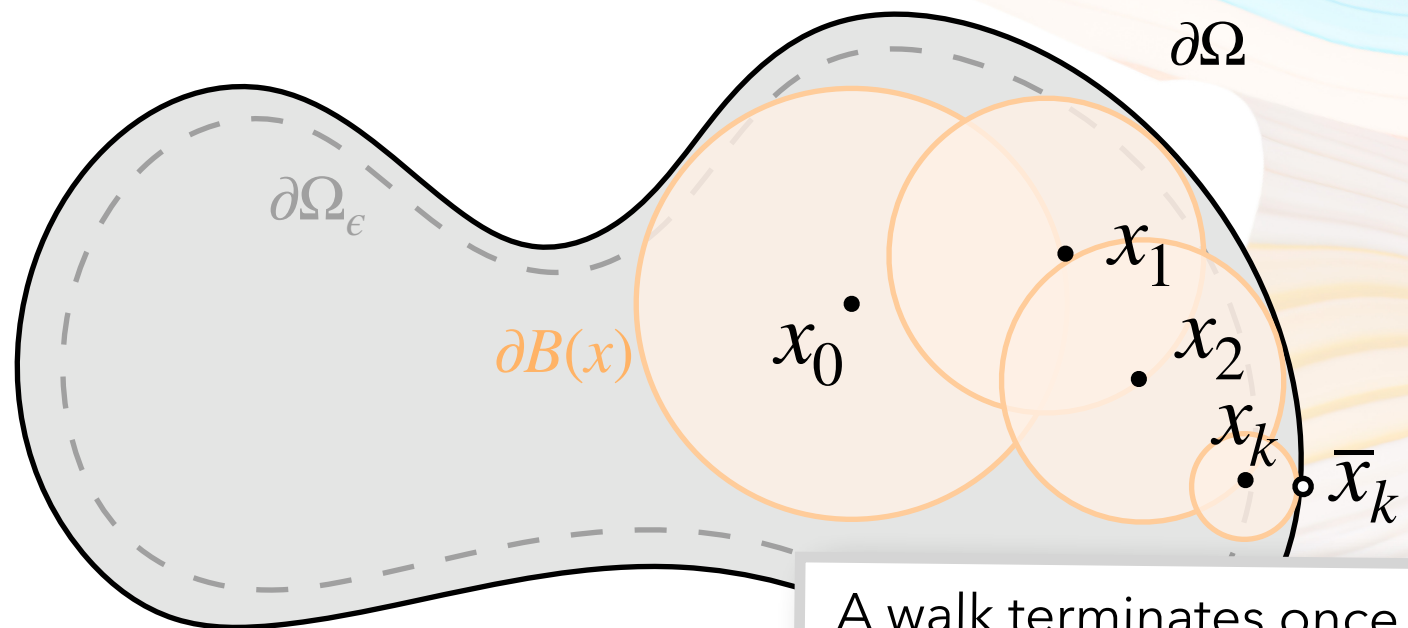
$$x_{i+1} \sim \mathcal{U} [\partial B(x_i)]$$



# walk on spheres [Muller 1956, Sawhney and Crane 2020]

$$\hat{u}(x_i) = \begin{cases} \hat{g}(x_{i+1}) & \text{if } x_{i+1} \in \partial\Omega_\epsilon \\ \hat{u}(x_{i+1}) & \text{otherwise} \end{cases}$$

$$x_{i+1} \sim \mathcal{U} [\partial B(x_i)]$$



A walk terminates once solution can be approximated with boundary data

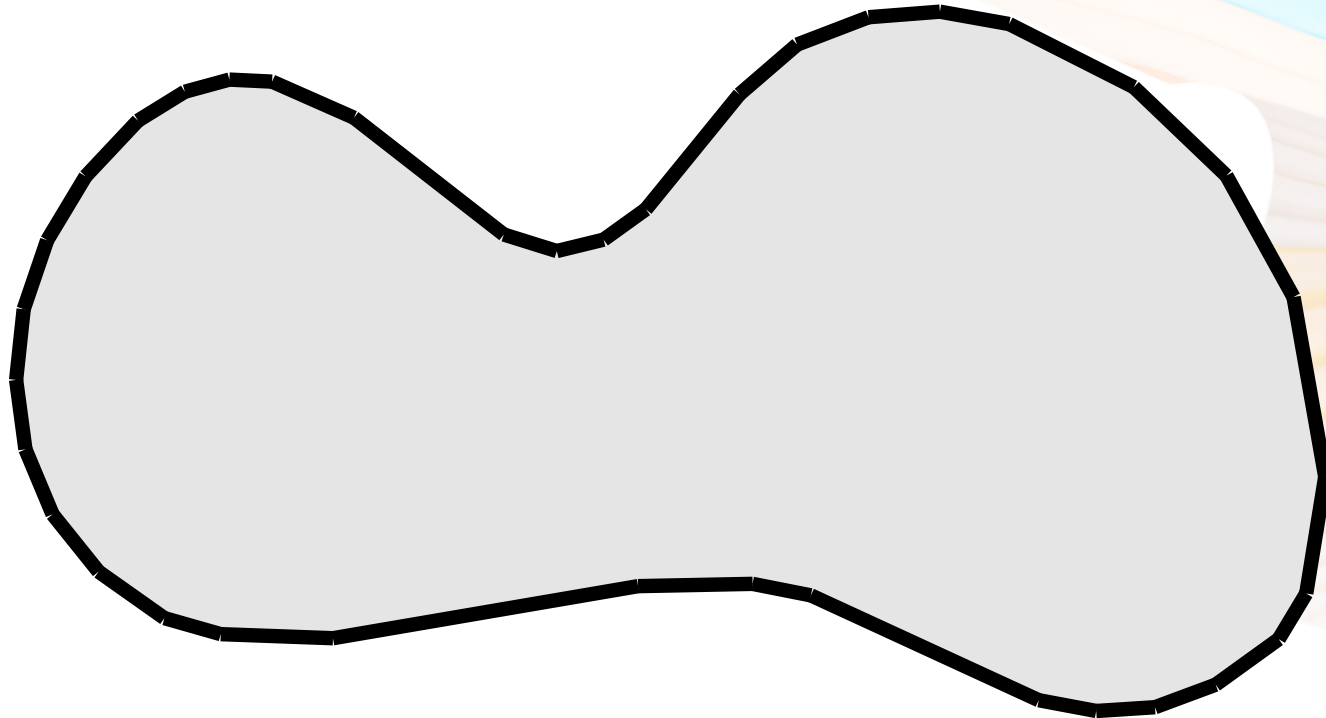
# differential walk on spheres



# differential quantities of a PDE

$$\Delta u = 0 \quad \text{on } \Omega$$

$$u = g \quad \text{on } \partial\Omega$$



# PDE with parameterized boundary

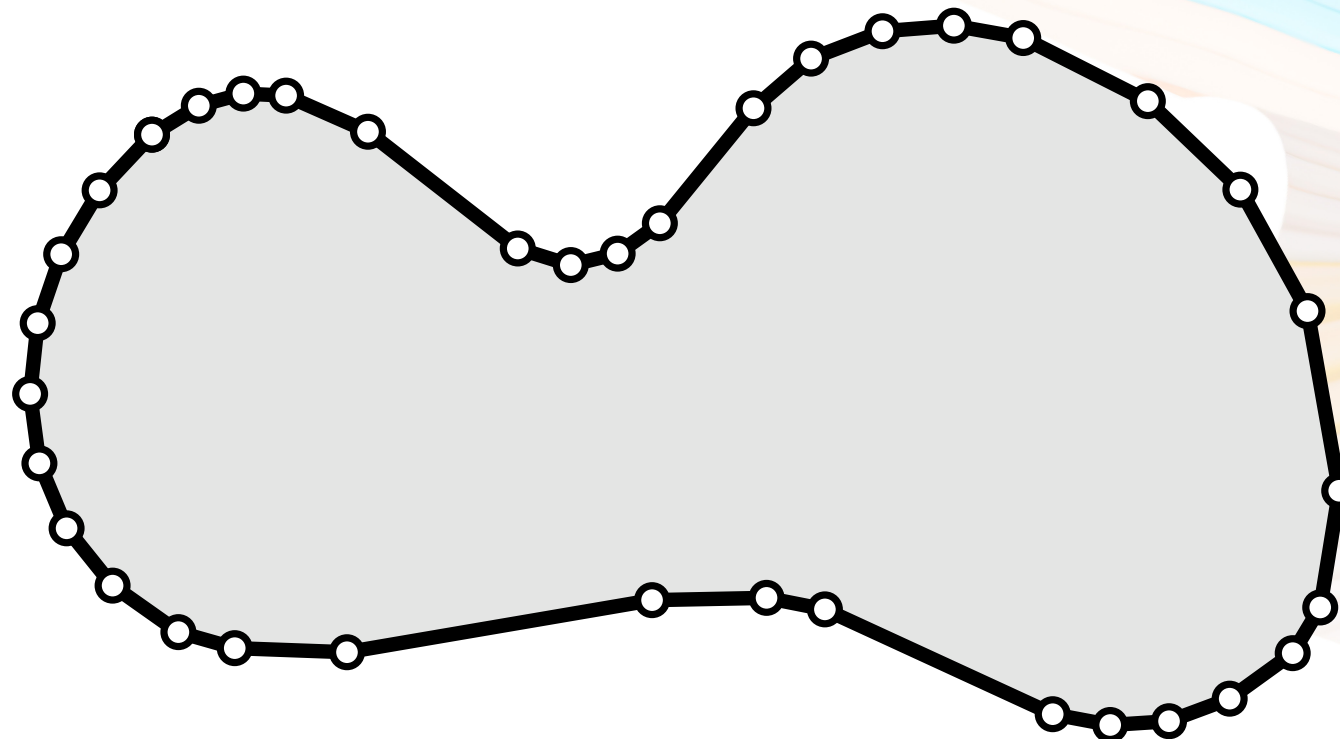
$$\Delta u = 0 \quad \text{on } \Omega(\pi)$$

$$u = g(x, \pi) \quad \text{on } \partial\Omega(\pi)$$

boundary parameters

$$\pi = [p_x^1, p_y^1, \dots, p_x^n, p_y^n]$$

$$v_i = (p_x^i, p_y^i) \quad (\text{vertex position})$$





# differential quantities of a PDE

$$\Delta u = 0 \quad \text{on } \Omega(\pi)$$

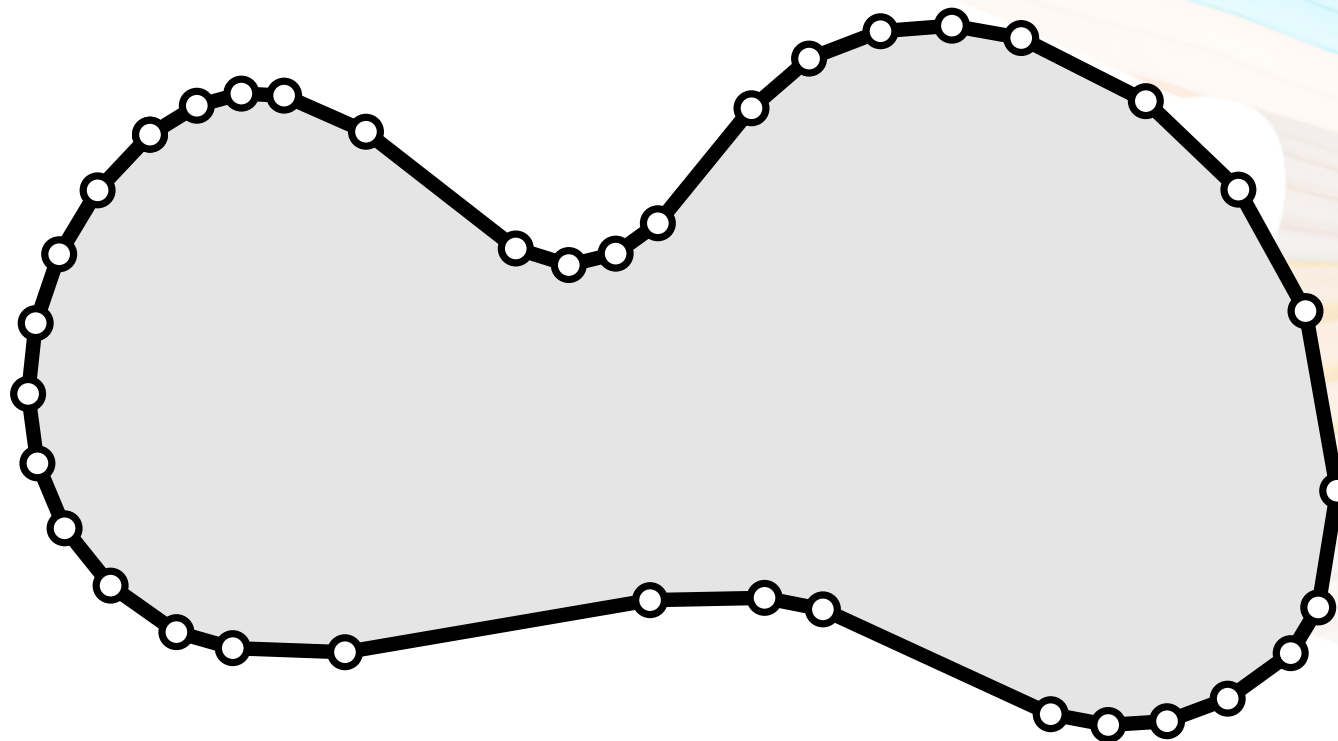
$$u = g \quad \text{on } \partial\Omega(\pi)$$

$$\dot{u} = \frac{\partial u}{\partial \pi}$$

boundary parameters

$$\pi = [p_x^1, p_y^1, \dots, p_x^n, p_y^n]$$

$$v_i = (p_x^i, p_y^i) \text{ (vertex position)}$$



# differential quantities of a PDE

$$\Delta u = 0 \quad \text{on } \Omega(\pi)$$

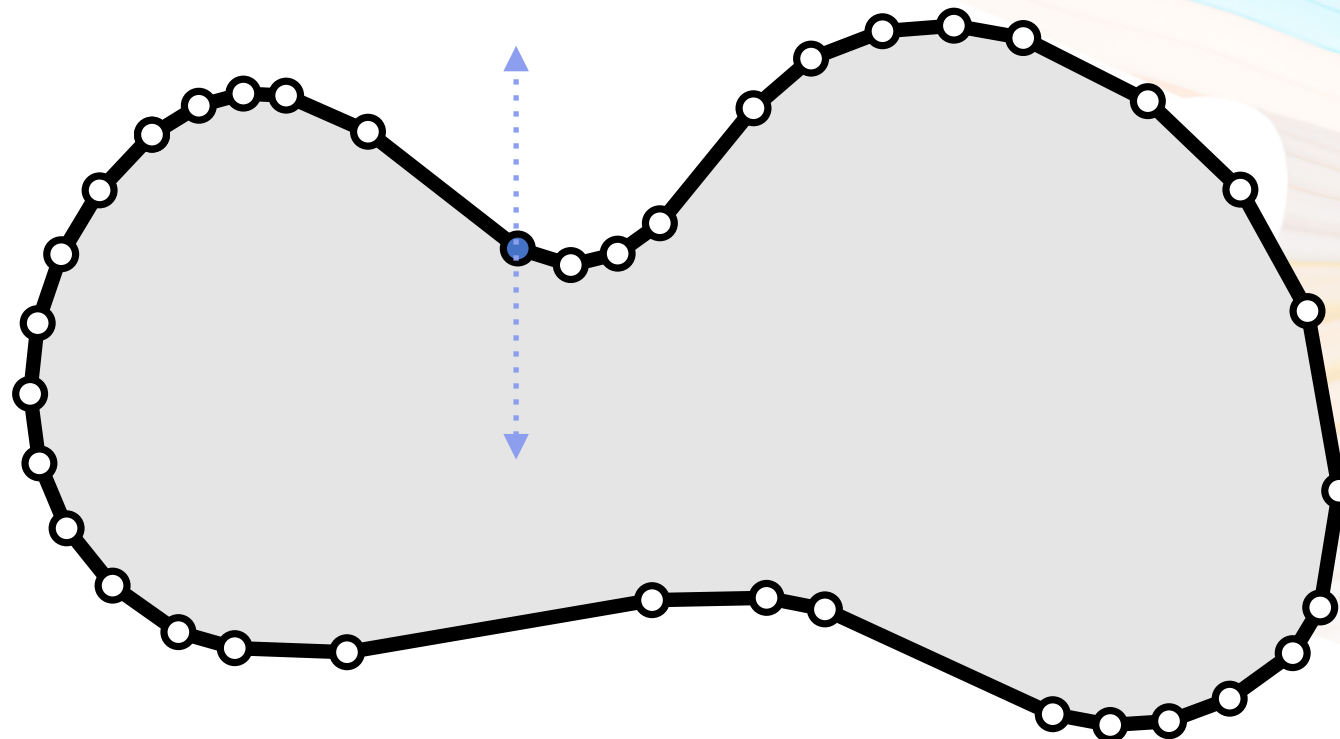
$$u = g \quad \text{on } \partial\Omega(\pi)$$

$$\dot{u} = \frac{\partial u}{\partial p_y^k}$$

boundary parameters

$$\pi = \begin{bmatrix} p_x^y \\ p_y^k \end{bmatrix}$$

$$v_i = (p_x^i, p_y^i) \text{ (vertex position)}$$



# differential quantities of a PDE

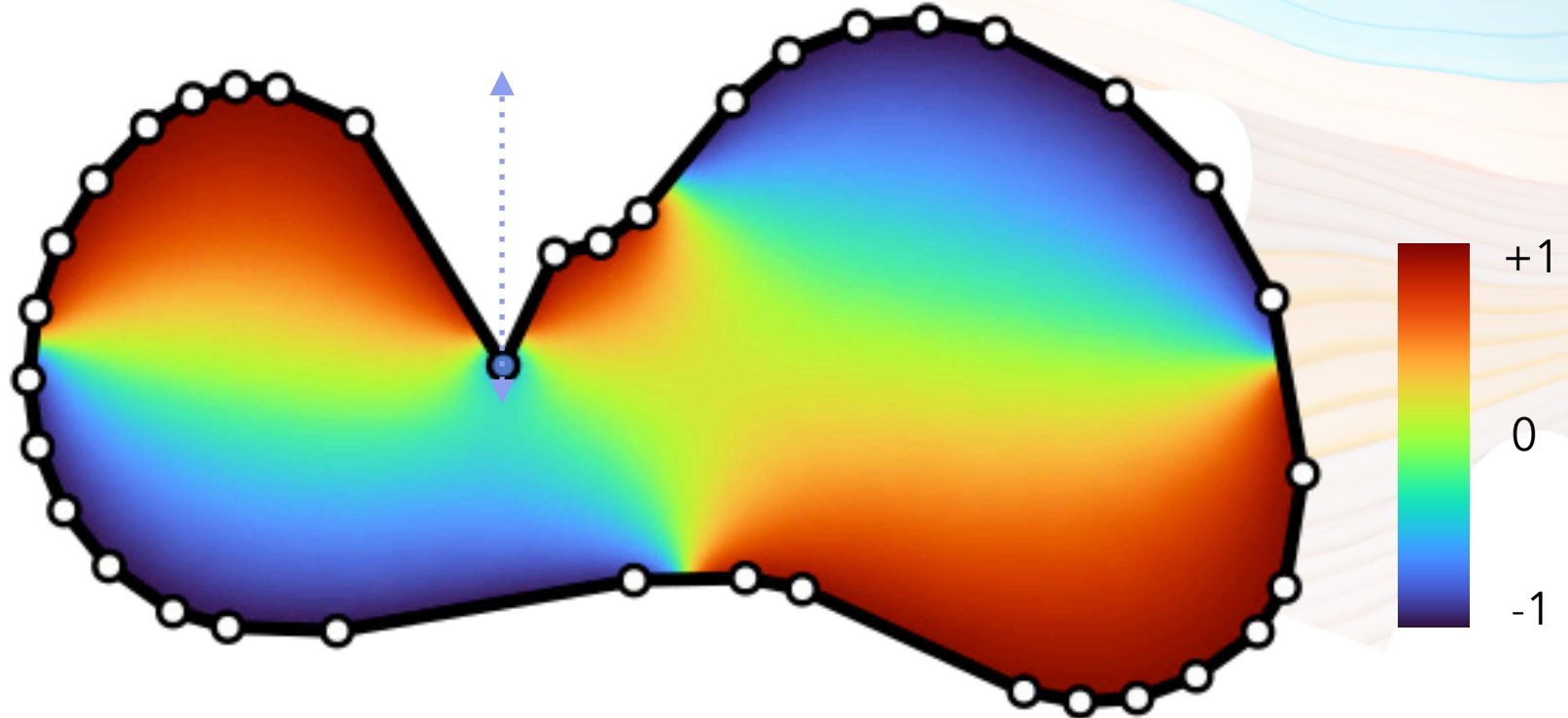
$$\begin{aligned} \Delta u &= 0 && \text{on } \Omega(\pi) \\ u &= g && \text{on } \partial\Omega(\pi) \end{aligned}$$

$$\dot{u} = \frac{\partial u}{\partial p_y^k}$$

boundary parameters

$$\pi = [p_k^y]$$

$$v_i = (p_x^i, p_y^i) \text{ (vertex position)}$$



# differential quantities of a PDE

$$\Delta u = 0 \quad \text{on } \Omega(\pi)$$

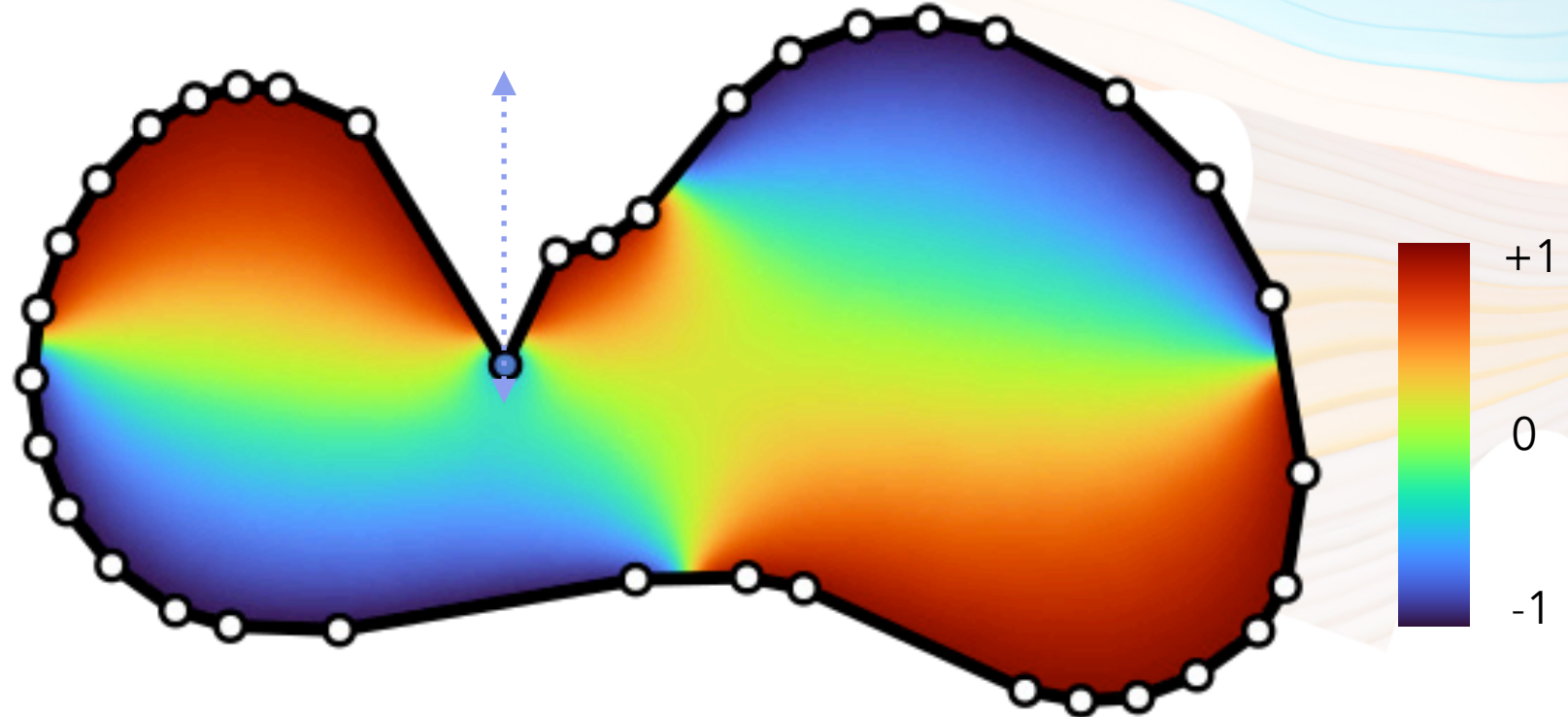
$$u = g \quad \text{on } \partial\Omega(\pi)$$

$$\dot{u} = \frac{\partial u}{\partial p_y^k}$$

boundary parameters

$$\pi = [p_k^y]$$

$$v_i = (p_x^i, p_y^i) \text{ (vertex position)}$$



# differential quantities of a PDE

$$\Delta u = 0 \quad \text{on } \Omega(\pi)$$

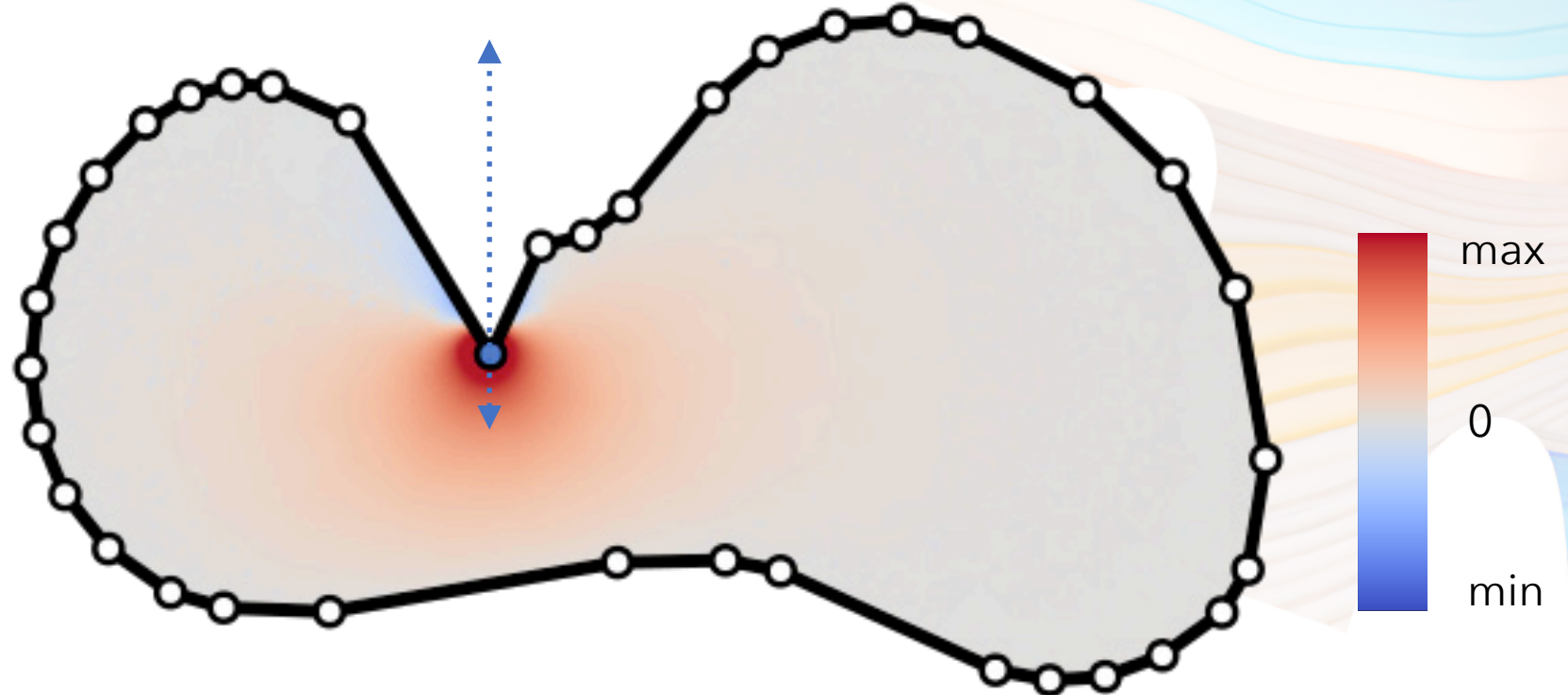
$$u = g \quad \text{on } \partial\Omega(\pi)$$

$$\dot{u} = \frac{\partial u}{\partial p_y^k}$$

boundary parameters

$$\pi = [p_k^y]$$

$$v_i = (p_x^i, p_y^i) \text{ (vertex position)}$$



# differential quantities of a PDE

$$\Delta u = 0 \quad \text{on } \Omega(\pi)$$

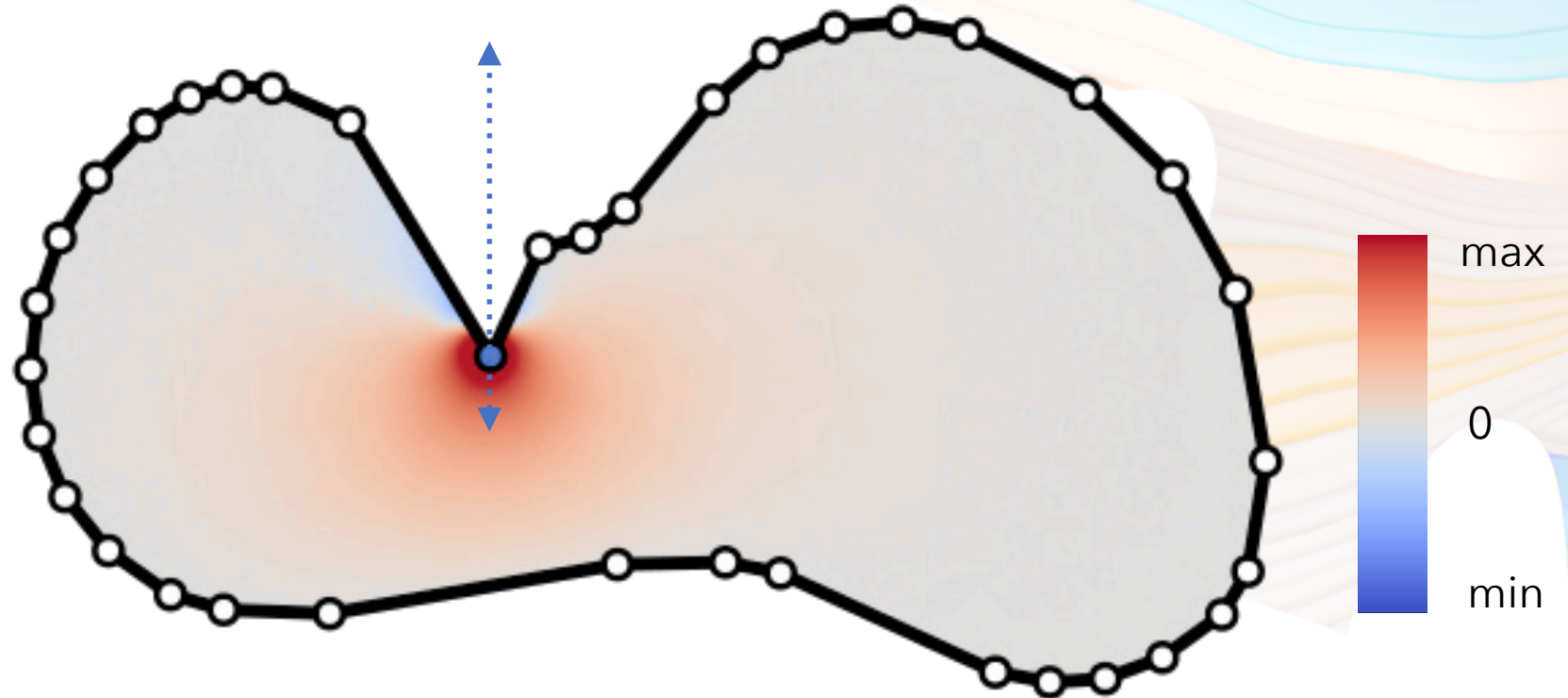
$$u = g \quad \text{on } \partial\Omega(\pi)$$

$$\dot{u} = \frac{\partial u}{\partial p_y^k}$$

boundary parameters

$$\pi = [p_k^y]$$

$$v_i = (p_x^i, p_y^i) \quad (\text{vertex position})$$



# differential quantities of a PDE

$$\Delta u = 0 \quad \text{on } \Omega(\pi)$$

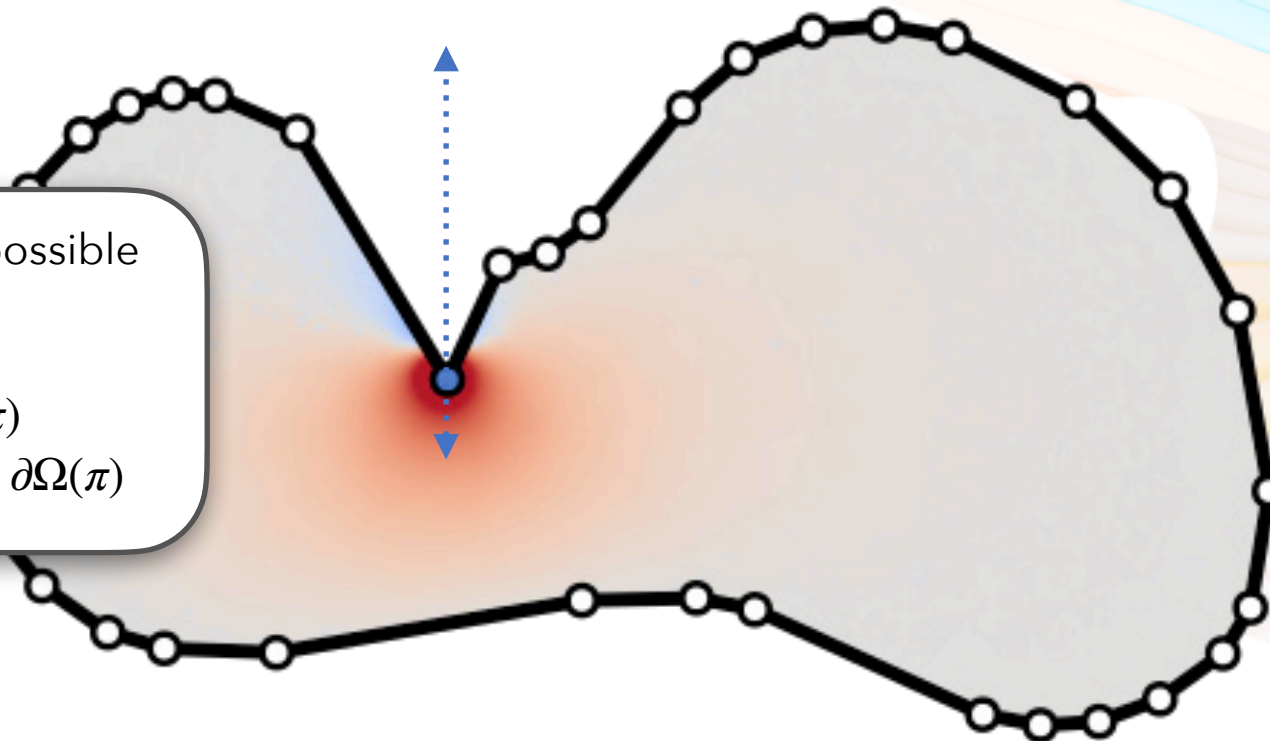
$$u = g \quad \text{on } \partial\Omega(\pi)$$

$$\dot{u} = \frac{\partial u}{\partial p_y^k}$$

boundary parameters

Differentiating **boundary data** also possible  
more complex PDEs

$$\begin{aligned} \nabla \alpha(x) \nabla u(x) - c(x) &= f(x) & \text{on } \Omega(\pi) \\ u(x) &= g(x, \pi) & \text{on } \partial\Omega(\pi) \end{aligned}$$



# differential quantities of a PDE

$$\Delta u = 0 \quad \text{on } \Omega(\pi)$$

$$u = g \quad \text{on } \partial\Omega(\pi)$$

$$\dot{u} = \frac{\partial u}{\partial p_y^k}$$

boundary parameters

Differentiating **boundary data** also possible  
more complex PDEs

$$\nabla \alpha(x) \nabla u(x) - c(x) = f(x) \quad \text{on } \Omega(\pi)$$

$$u(x) = g(x, \pi) \quad \text{on } \partial\Omega(\pi)$$

Differentiating **volumetric data** covered by  
Yilmazer et al. later in this session!

$$\nabla \alpha(x, \pi) \nabla u(x) - c(x, \pi) = f(x, \pi) \quad \text{on } \Omega$$

$$u(x) = g(x) \quad \text{on } \partial\Omega$$





# deriving differential Monte Carlo solver

mean value integral

$$u(x) = \frac{1}{|\partial B(x)|} \int_{\partial B(x)} u(y) dy$$

# deriving differential Monte Carlo solver

mean value integral

$$u(x) = \frac{1}{|\partial B(x)|} \int_{\partial B(x)} u(y) dy$$

differentiate the mean value integral ?

$$\frac{\partial}{\partial \pi} u(x) = \frac{\partial}{\partial \pi} \left[ \frac{1}{|\partial B(x)|} \int_{\partial B(x)} u(y) dy \right]$$

# deriving differential Monte Carlo solver

concurrent work pursues this type of approach

mean value integral

$$u(x) = \frac{1}{|\partial B(x)|} \int_{\partial B(x)} u(y) dy$$

differentiate the mean value integral ?

$$\frac{\partial}{\partial \pi} u(x) = \frac{\partial}{\partial \pi} \left[ \frac{1}{|\partial B(x)|} \int_{\partial B(x)} u(y) dy \right]$$

### A Differential Monte Carlo Solver For the Poisson Equation

Zihan Yu zhiyu19@uci.edu  
University of California, Irvine & NVIDIA USA

Lifan Wu lifanw@nvidia.com  
NVIDIA USA

Zhiqian Zhou zhiqiaz28@uci.edu  
University of California, Irvine USA

Shuang Zhao shz@ics.uci.edu  
University of California, Irvine & NVIDIA USA

Figure 1: We introduce a new grid-free technique to estimate derivatives of solutions to the Poisson equation with respect to arbitrary parameters including domain shapes. This example includes a 3D Laplace problem with Dirichlet boundary conditions on a wired bunny shape. We visualize the solution to this problem in two cross-sectional planes in (a) and the derivative of this solution (with respect to the translation of the bunny) estimated with our method in (b).

**ABSTRACT**  
The Poisson equation is an important partial differential equation (PDE) with numerous applications in physics, engineering, and computer graphics. Conventional solutions to the Poisson equation require discretizing the domain or its boundary, which can be very expensive for domains with detailed geometries. To overcome this challenge, a family of grid-free Monte Carlo solutions has recently been developed. By utilizing walk-on-sphere (WoS) processes, these techniques are capable of efficiently solving the Poisson equation over complex domains.

In this paper, we introduce a general technique that differentiates solutions to the Poisson equation with Dirichlet boundary conditions. Specifically, we devise a new boundary-integral formulation for the derivatives with respect to arbitrary parameters including shapes of the domain. Further, we develop an efficient walk-on-spheres technique based on our new formulation—including a new approach to estimate normal derivatives of the solution field. We demonstrate the effectiveness of our technique over baseline methods using several synthetic examples.

**CCS CONCEPTS**  
• Mathematics of computing → Partial differential equations; Integral equations; Probabilistic algorithms.

[Yu et al. 2024]

### Solving Inverse PDE Problems using Grid-Free Monte Carlo Estimators

EKREM FATIH YILMAZER, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland  
DELIO VICINI, Google Inc., Switzerland  
WENZEL JAKOB, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Figure 1: We apply our inverse PDE solver to a 2D electrical impedance tomography experiment [Hauptmann et al. 2017], in which an electric current flows through a saline-filled water tank containing conducting objects of different sizes (photographs in middle). The marker in the middle indicates the center. Injecting a current using 2 of its uniformly spaced electrodes at the tank boundary generates a measurable voltage difference at the other electrodes, and injecting at various locations produces a matrix  $u_{ref}$  of measurements. The objective of this inverse problem is to infer the properties of the conductor from this data. We perform a differentiable simulation of this setup to optimize the center and radius of a conducting circle. The frames on the left show the progression of the optimization (columns (a)), while the rightmost two columns reveal how the predicted voltages become increasingly consistent with the measurement (columns (d)).

Partial differential equations can model diverse physical phenomena including heat diffusion, incompressible flows, and electrostatic potentials. Given a description of an object's boundary and interior, traditional methods solve such PDEs by densely meshing the interior and then solving a large and sparse linear system derived from this mesh. Recent grid-free solvers take an alternative approach and avoid this complexity in exchange for randomness: they compute stochastic solution estimates and generally bear a striking resemblance to physically-based rendering algorithms.

In this article, we develop algorithms targeting the inverse form of this problem: given an already existing solution of a PDE, we infer parameters characterizing the boundary and interior. In the grid-free setting, there are again significant connections to rendering, and we show how insights from both fields can be combined to compute unbiased derivative estimates that enable gradient-based optimization. In this process, we encounter new challenges that must be addressed to obtain practical solutions. We introduce acceleration and variance reduction strategies and show how to differentiate branching random walks in reverse mode.

We finally demonstrate our approach on both simulated data and a real-world electrical impedance tomography experiment, where we reconstruct the position of a conducting object from voltage measurements taken in a saline-filled tank.

**CCS Concepts** • Mathematics of computing → Partial differential equations; • Computing methodologies → Rendering

**Additional Key Words and Phrases:** Walk on Spheres, Differentiable Rendering, Path Reply Backpropagation, Electrical Impedance Tomography

**ACM Reference Format:**  
Ekrem Fatih Yilmazer, Delio Vicini, and Wenzel Jakob. 2024. Solving Inverse PDE Problems using Grid-Free Monte Carlo Estimators. *ACM Trans. Graph.* 43, 6, Article 175 (December 2024), 18 pages. <https://doi.org/10.1145/3687990>

**1 INTRODUCTION**  
Many physical phenomena are naturally described using partial differential equations (PDEs). For example, the heat equation models the spread of thermal energy in a potentially heterogeneous material. Solvers that numerically approximate solutions of such PDEs are in widespread use. We pursue the opposite direction in this article, which is known as an *inverse PDE problem*: estimating unknown parameters from observations of the solution. This set of unknown parameters could include various PDE coefficients, boundary conditions, and even the shape of the domain.

*ACM Trans. Graph.*, Vol. 43, No. 6, Article 175. Publication date: December 2024.

[Yilmazer et al. 2024]

# deriving differential Laplace PDE

primal PDE

$$\begin{aligned}\Delta u &= 0 && \text{on } \Omega(\pi) \\ u &= g && \text{on } \partial\Omega(\pi)\end{aligned}$$

differential PDE

$$\begin{aligned}\Delta \dot{u} &= 0 && \text{on } \Omega(\pi) \\ \dot{u} &= V_n \left( \frac{\partial g}{\partial n} - \frac{\partial u}{\partial n} \right) && \text{on } \partial\Omega(\pi)\end{aligned}$$

# deriving differential Laplace PDE

primal PDE

$$\begin{aligned}\Delta u &= 0 && \text{on } \Omega(\pi) \\ u &= g && \text{on } \partial\Omega(\pi)\end{aligned}$$

differential PDE

$$\begin{aligned}\Delta \dot{u} &= 0 && \text{on } \Omega(\pi) \\ \dot{u} &= V_n \left( \frac{\partial g}{\partial n} - \frac{\partial u}{\partial n} \right) && \text{on } \partial\Omega(\pi)\end{aligned}$$

**classic result from shape optimization**

details in Henrot and Pierre [2018]

# deriving differential Laplace PDE

primal PDE

$$\begin{aligned} \Delta u &= 0 && \text{on } \Omega(\pi) \\ u &= g && \text{on } \partial\Omega(\pi) \end{aligned}$$

differential PDE

$$\begin{aligned} \Delta \dot{u} &= 0 && \text{on } \Omega(\pi) \\ \dot{u} &= V_n \left( \frac{\partial g}{\partial n} - \frac{\partial u}{\partial n} \right) && \text{on } \partial\Omega(\pi) \end{aligned}$$

**classic result from shape optimization**

details in Henrot and Pierre [2018]

Change in the  
shape of the  
boundary

$$V = \frac{\partial \Phi(x, \pi)}{\partial \pi}$$

# deriving differential Laplace PDE

primal PDE

$$\begin{aligned} \Delta u &= 0 && \text{on } \Omega(\pi) \\ u &= g && \text{on } \partial\Omega(\pi) \end{aligned}$$

differential PDE

$$\begin{aligned} \Delta \dot{u} &= 0 && \text{on } \Omega(\pi) \\ \dot{u} &= V_n \left( \frac{\partial g}{\partial n} - \frac{\partial u}{\partial n} \right) && \text{on } \partial\Omega(\pi) \end{aligned}$$

**classic result from shape optimization**

details in Henrot and Pierre [2018]

Change in the  
shape of the  
boundary

Change in  
boundary  
condition

$$V = \frac{\partial \Phi(x, \pi)}{\partial \pi} \quad g : \mathbb{R}^3 \rightarrow \mathbb{R}$$

# deriving differential Laplace PDE

primal PDE

$$\begin{aligned} \Delta u &= 0 && \text{on } \Omega(\pi) \\ u &= g && \text{on } \partial\Omega(\pi) \end{aligned}$$

differential PDE

$$\begin{aligned} \Delta \dot{u} &= 0 && \text{on } \Omega(\pi) \\ \dot{u} &= V_n \left( \frac{\partial g}{\partial n} - \frac{\partial u}{\partial n} \right) && \text{on } \partial\Omega(\pi) \end{aligned}$$

**classic result from shape optimization**

details in Henrot and Pierre [2018]

Change in the shape of the boundary  
 $V = \frac{\partial \Phi(x, \pi)}{\partial \pi}$

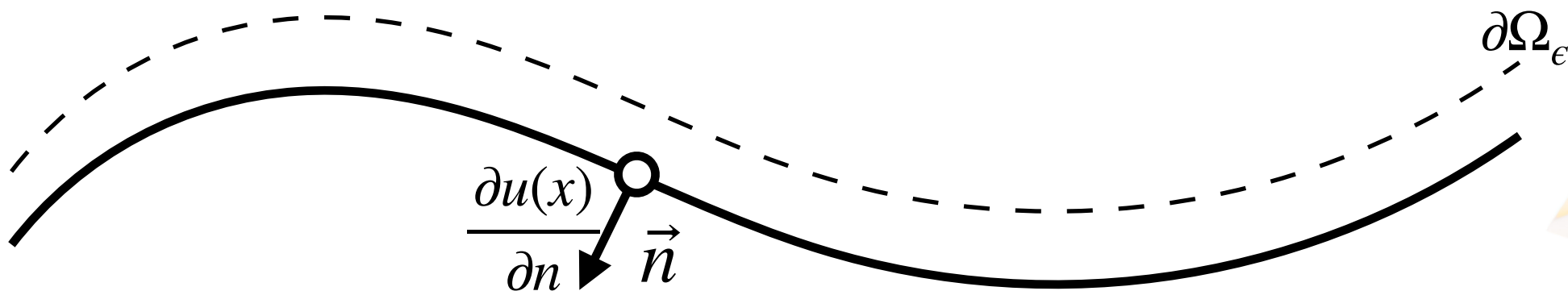
Change in boundary condition  
 $g : \mathbb{R}^3 \rightarrow \mathbb{R}$

Change in PDE solution



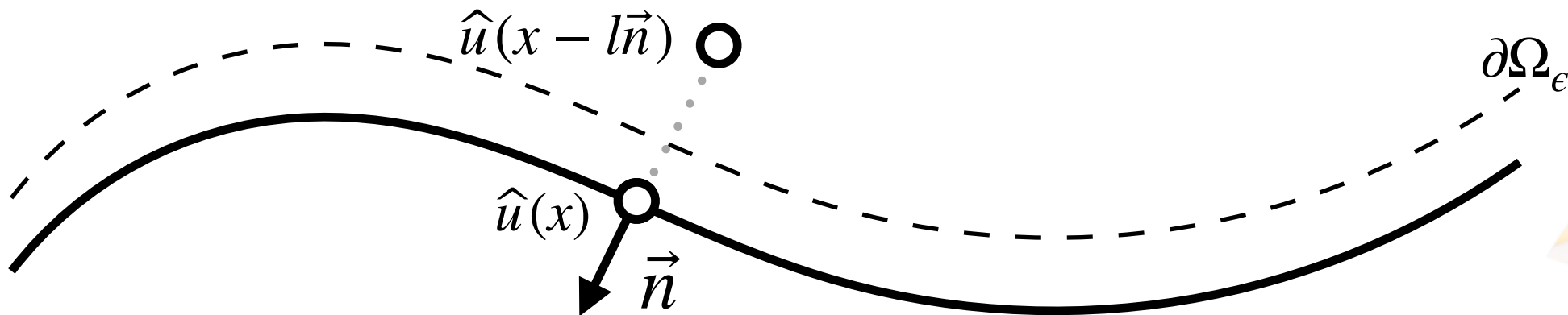
# estimating normal derivative at boundary

$$\frac{\widehat{\partial u}}{\partial n}(x) \approx \frac{\widehat{u}(x) - \widehat{u}(x - l\vec{n})}{l}$$



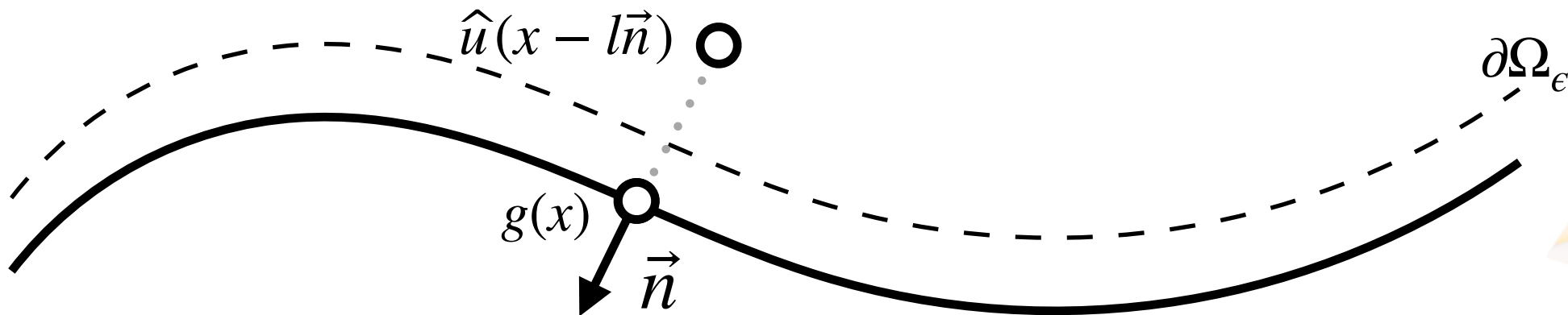
# estimating normal derivative at boundary

$$\frac{\widehat{\partial u}}{\partial n}(x) \approx \frac{\widehat{u}(x) - \widehat{u}(x - l\vec{n})}{l}$$



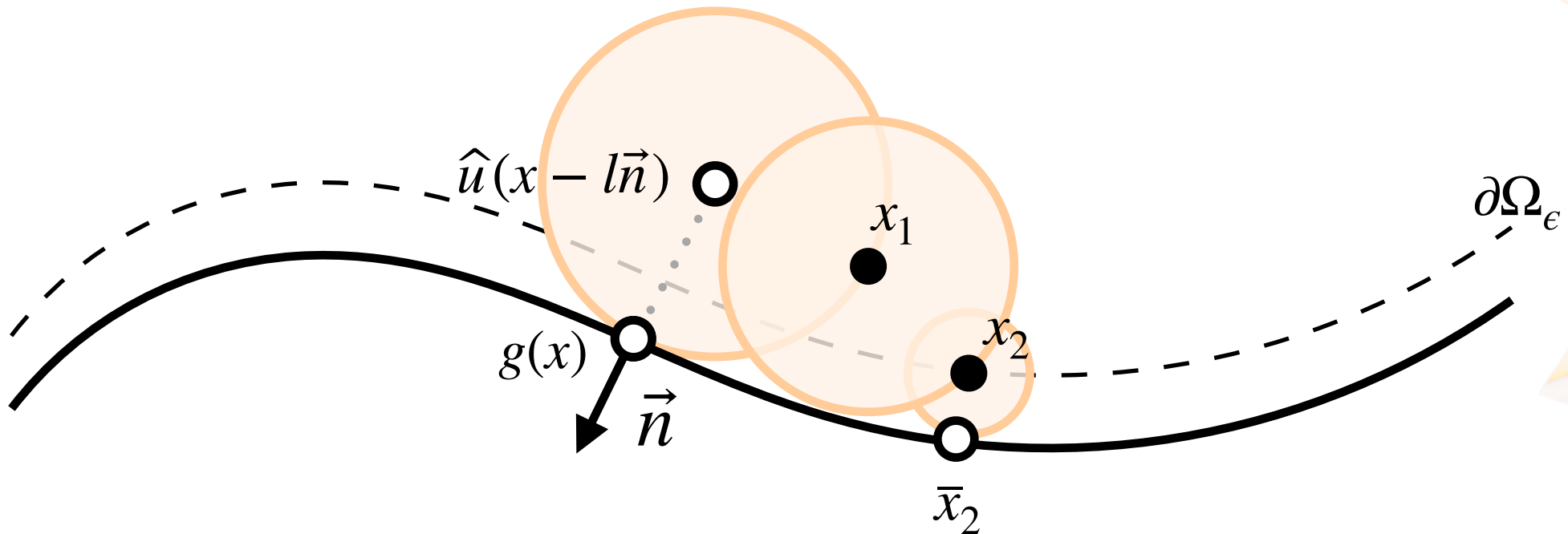
# estimating normal derivative at boundary

$$\frac{\widehat{\partial u}}{\partial n}(x) \approx \frac{g(x) - \widehat{u}(x - l\vec{n})}{l}$$



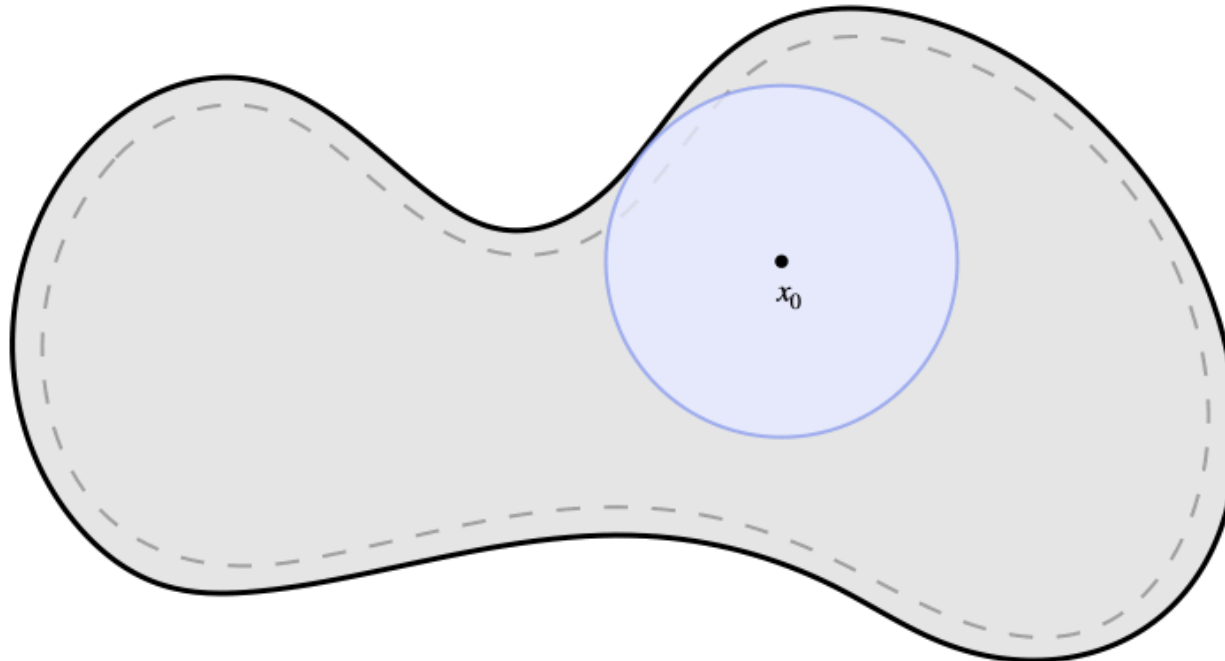
# estimating normal derivative at boundary

$$\frac{\widehat{\partial u}}{\partial n}(x) \approx \frac{g(x) - \widehat{u}(x - l\vec{n})}{l}$$



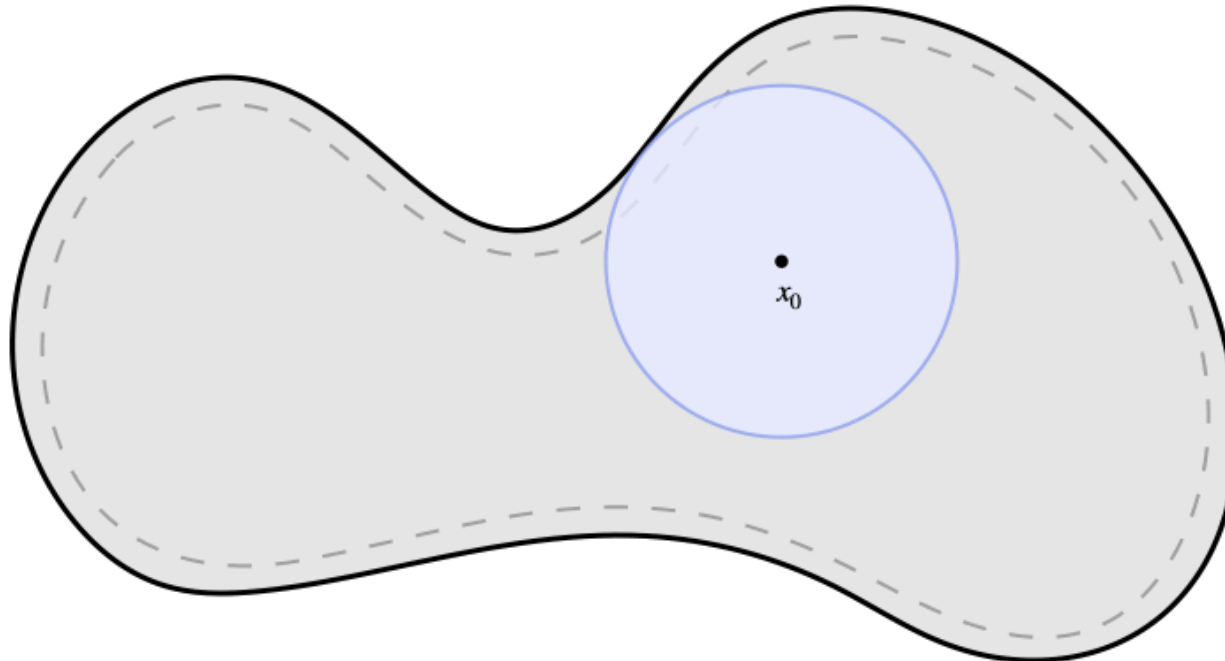
# solving differential PDE with walk on spheres

$$\begin{aligned} \Delta \dot{u} &= 0 && \text{on } \Omega(\pi) \\ \dot{u} &= V_n \left( \frac{\partial g}{\partial n} - \frac{\partial u}{\partial n} \right) && \text{on } \partial\Omega(\pi) \end{aligned}$$



# solving differential PDE with walk on spheres

$$\begin{aligned} \Delta \dot{u} &= 0 && \text{on } \Omega(\pi) \\ \dot{u} &= V_n \left( \frac{\partial g}{\partial n} - \frac{\partial u}{\partial n} \right) && \text{on } \partial\Omega(\pi) \end{aligned}$$

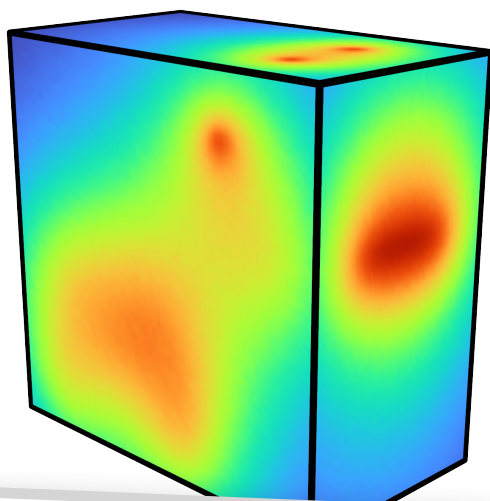
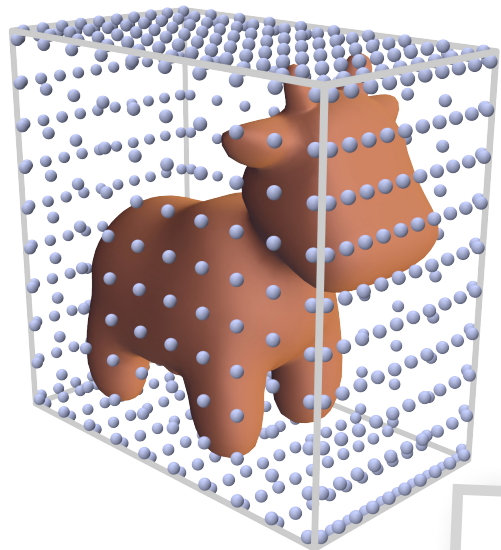


# inherits benefits of walk on spheres



# inherits benefits of walk on spheres

pointwise evaluation

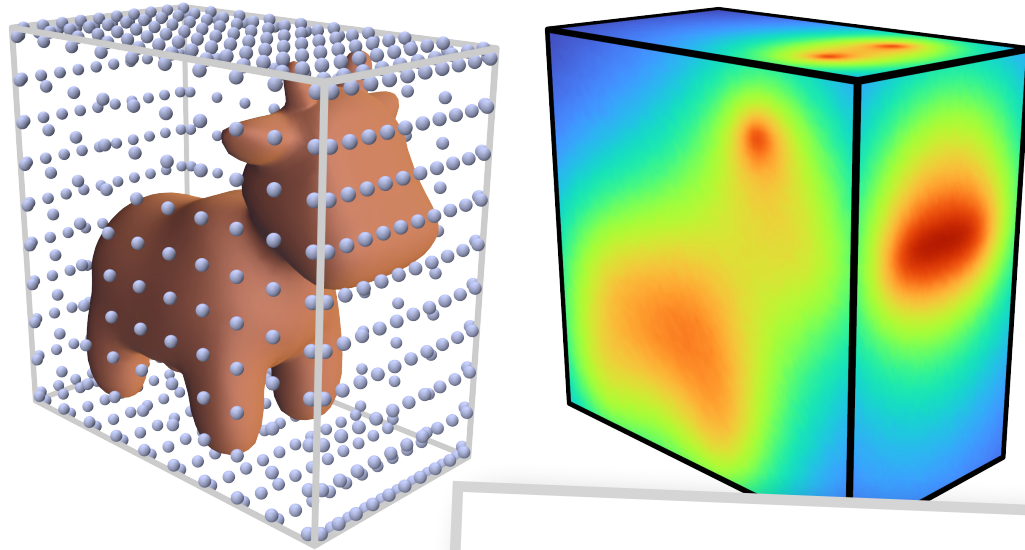


Evaluate derivatives  
only where needed



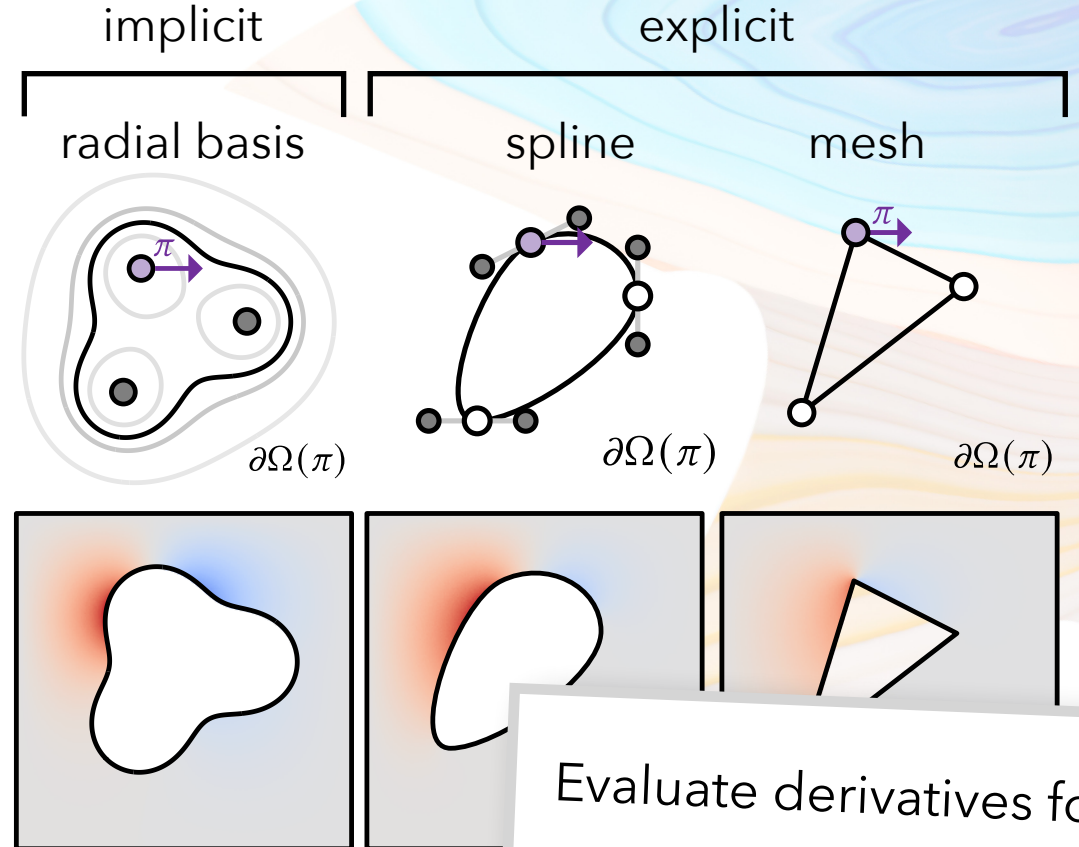
# inherits benefits of walk on spheres

## pointwise evaluation



Evaluate derivatives only where needed

## geometry representations

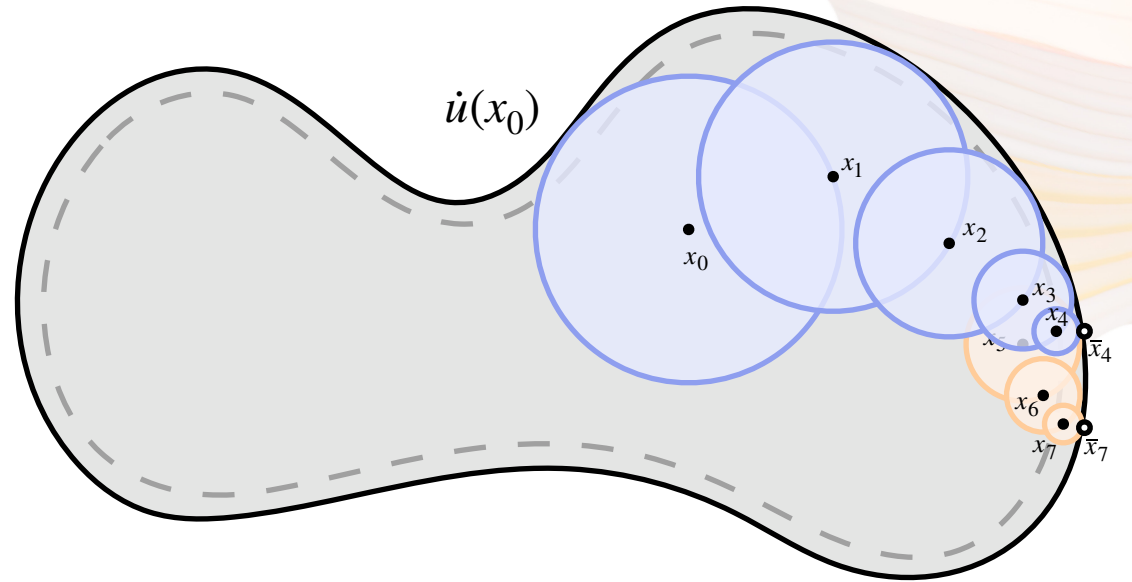
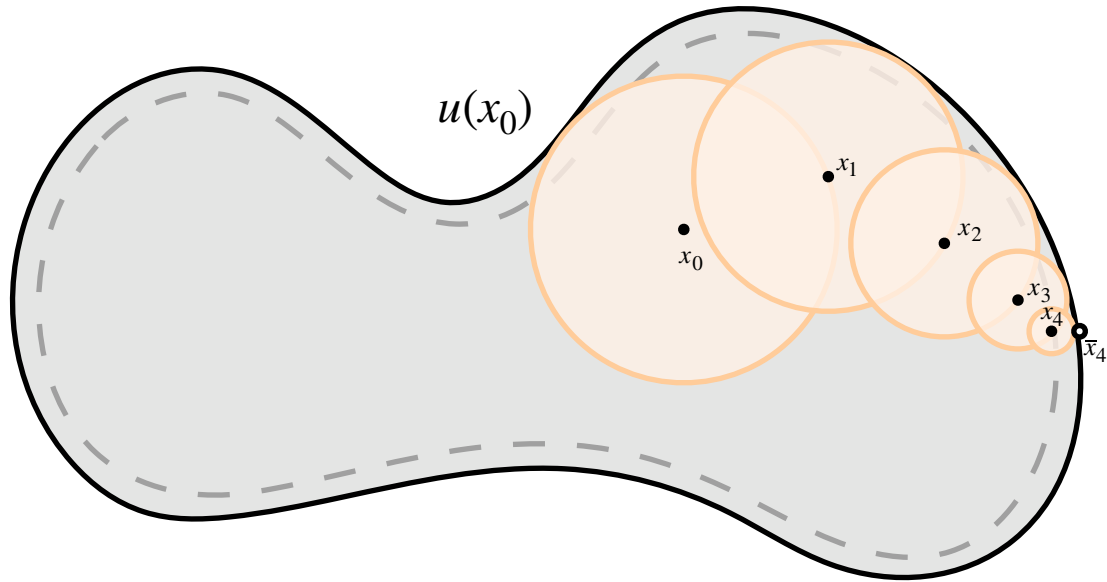


Evaluate derivatives for many representations

# sharing primary walk to improve performance

$$\begin{aligned} \Delta u &= 0 && \text{on } \Omega(\pi) \\ u &= g && \text{on } \partial\Omega(\pi) \end{aligned}$$

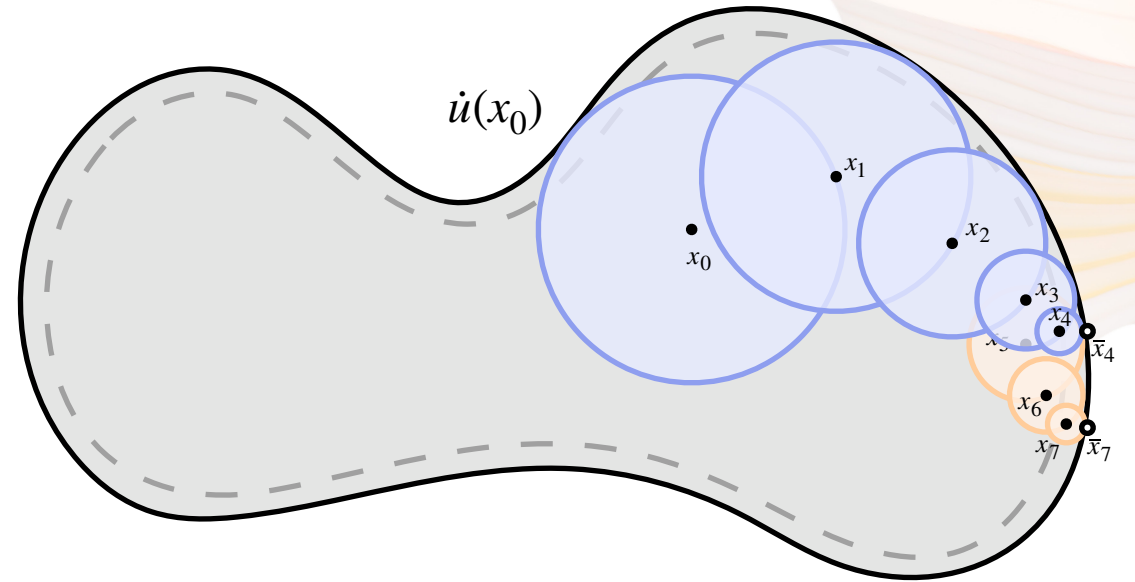
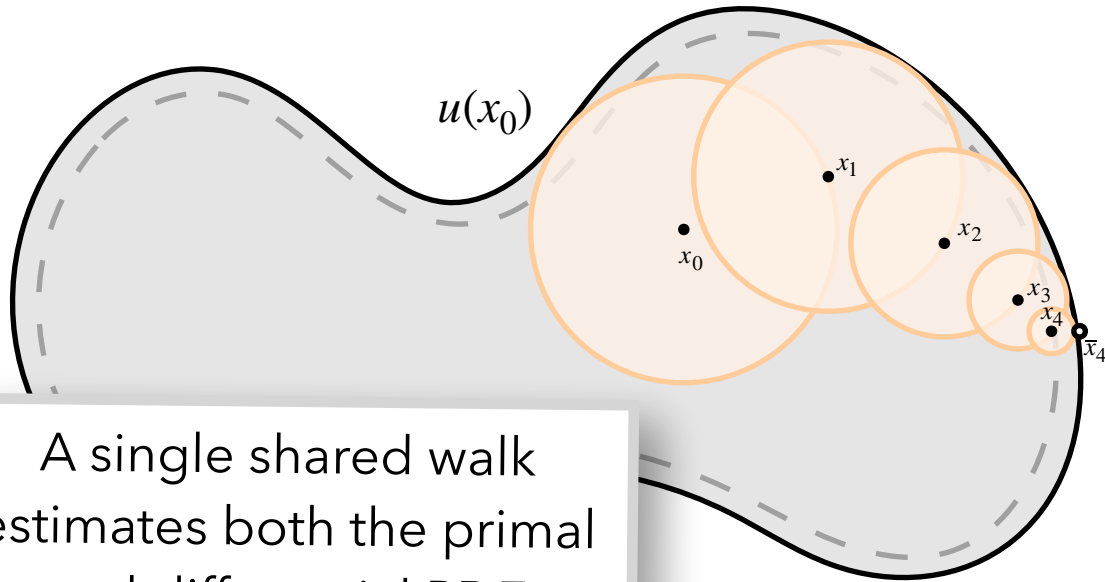
$$\begin{aligned} \Delta \dot{u} &= 0 && \text{on } \Omega(\pi) \\ \dot{u} &= V_n \left( \frac{\partial g}{\partial n} - \frac{\partial u}{\partial n} \right) && \text{on } \partial\Omega(\pi) \end{aligned}$$



# sharing primary walk to improve performance

$$\begin{aligned} \Delta u &= 0 && \text{on } \Omega(\pi) \\ u &= g && \text{on } \partial\Omega(\pi) \end{aligned}$$

$$\begin{aligned} \Delta \dot{u} &= 0 && \text{on } \Omega(\pi) \\ \dot{u} &= V_n \left( \frac{\partial g}{\partial n} - \frac{\partial u}{\partial n} \right) && \text{on } \partial\Omega(\pi) \end{aligned}$$



A single shared walk estimates both the primal and differential PDE

# scaling with number of parameters

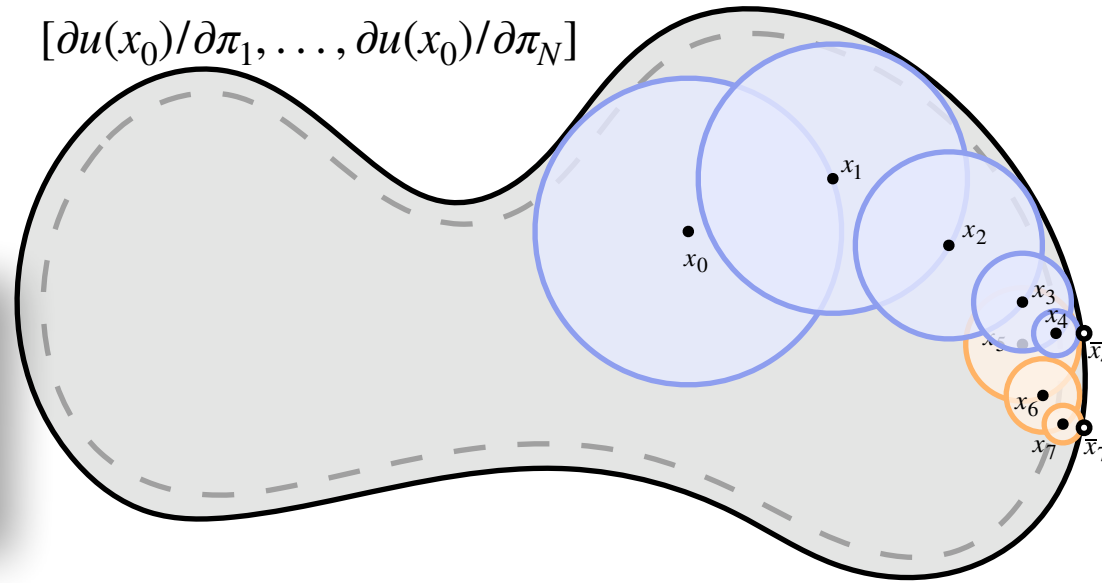
differential PDE over several parameters

$$\Delta[\partial u / \partial \pi_1, \dots, \partial u / \partial \pi_N] = 0$$

on  $\Omega(\pi)$

$$\underbrace{[\partial u / \partial \pi_1, \dots, \partial u / \partial \pi_N]}_{\dot{u}} = \underbrace{[n \cdot \partial \Phi / \partial \pi_1, \dots, n \cdot \partial \Phi / \partial \pi_N]}_{V_n} \left( \frac{\partial g}{\partial n} - \frac{\partial u}{\partial n} \right) \text{ on } \partial \Omega(\pi)$$

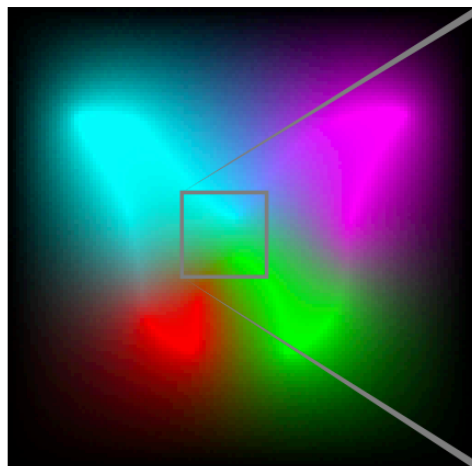
$$[\partial u(x_0) / \partial \pi_1, \dots, \partial u(x_0) / \partial \pi_N]$$



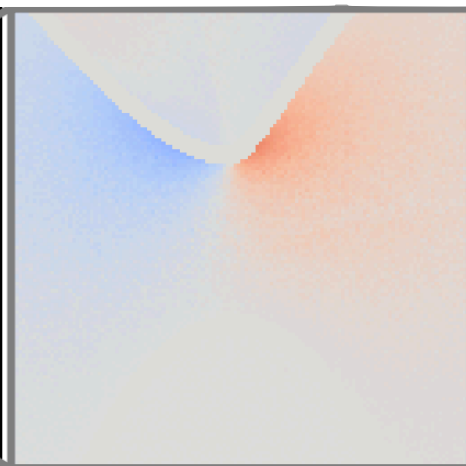
A single shared walk estimates derivatives for all parameters

# scaling with number of parameters

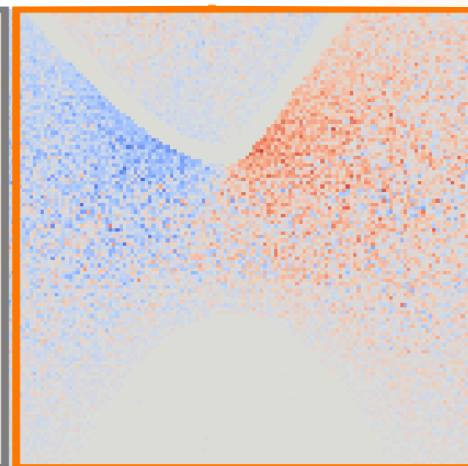
PDE solution



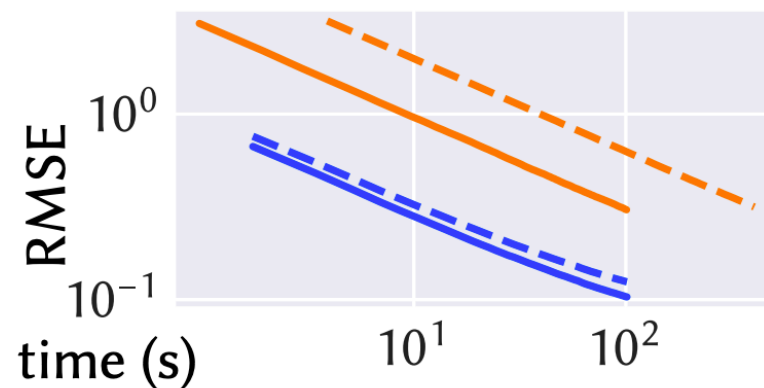
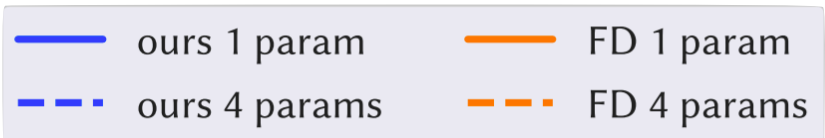
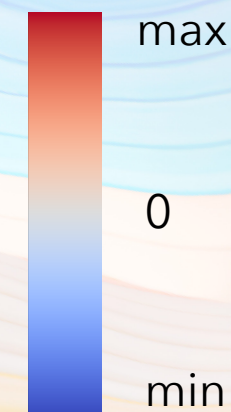
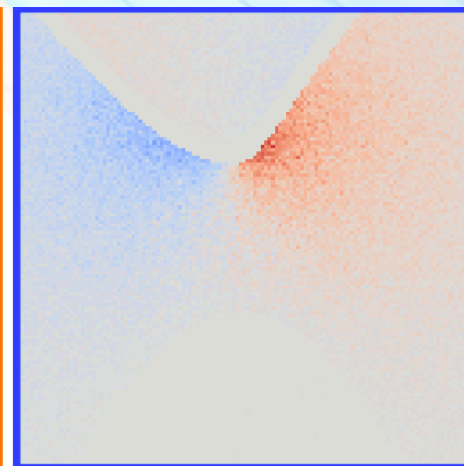
differential PDE reference



finite differences (equal time)

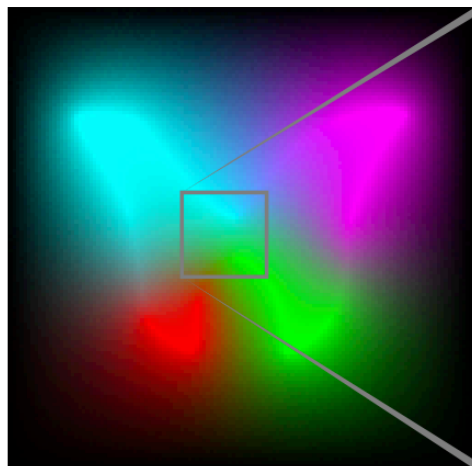


ours (equal time)

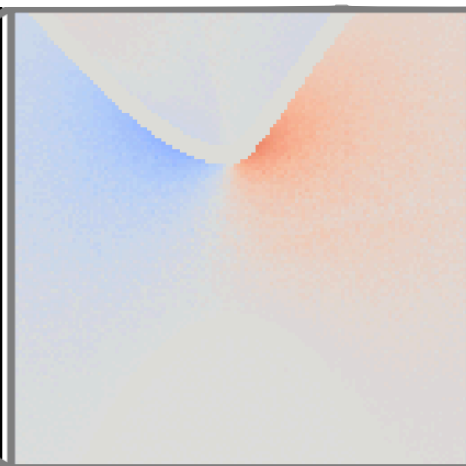


# scaling with number of parameters

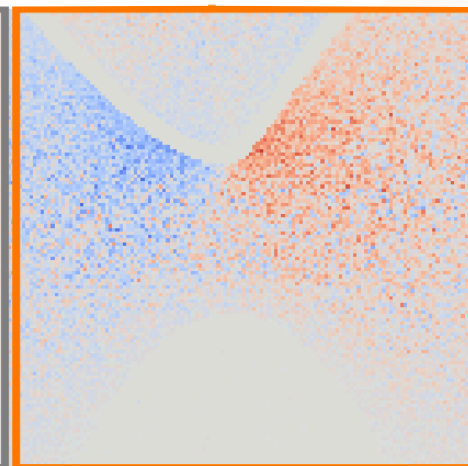
PDE solution



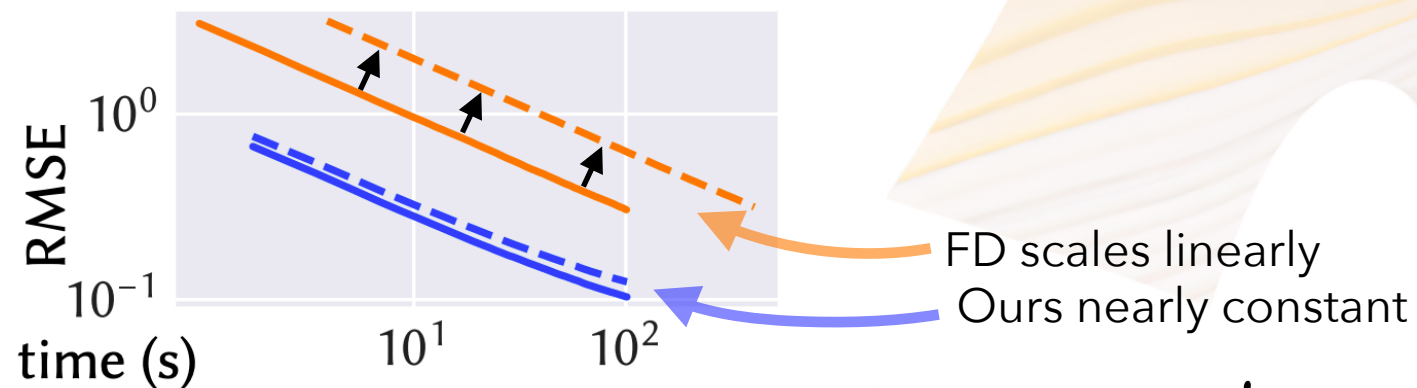
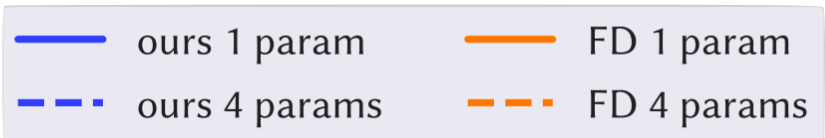
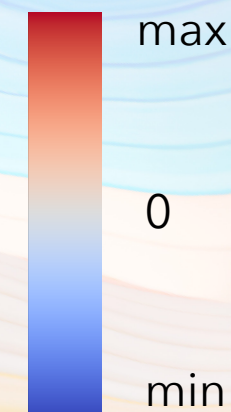
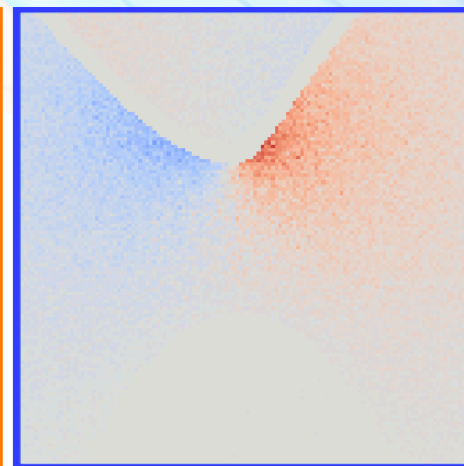
differential PDE reference



finite differences (equal time)

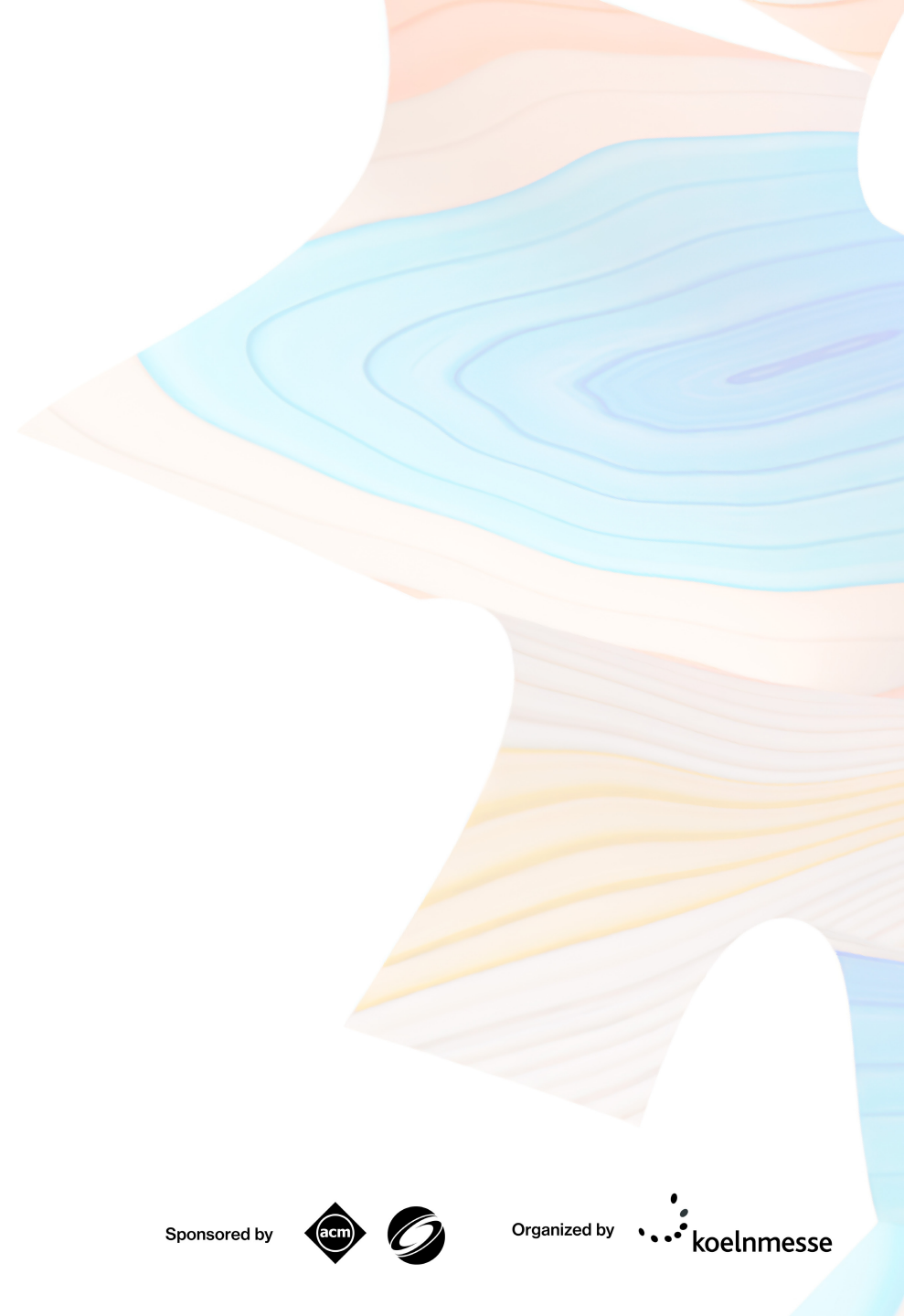
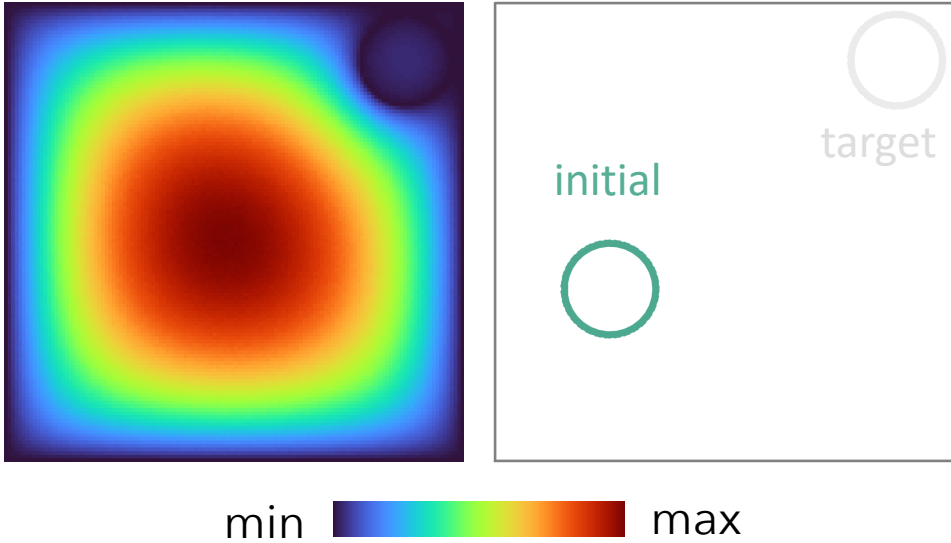


ours (equal time)



# stochastic gradients

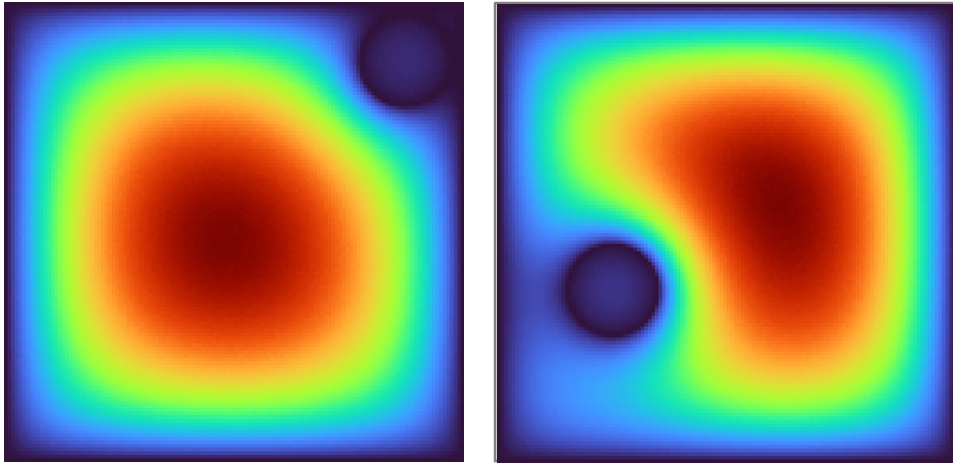
target PDE solution    initial geometry guess



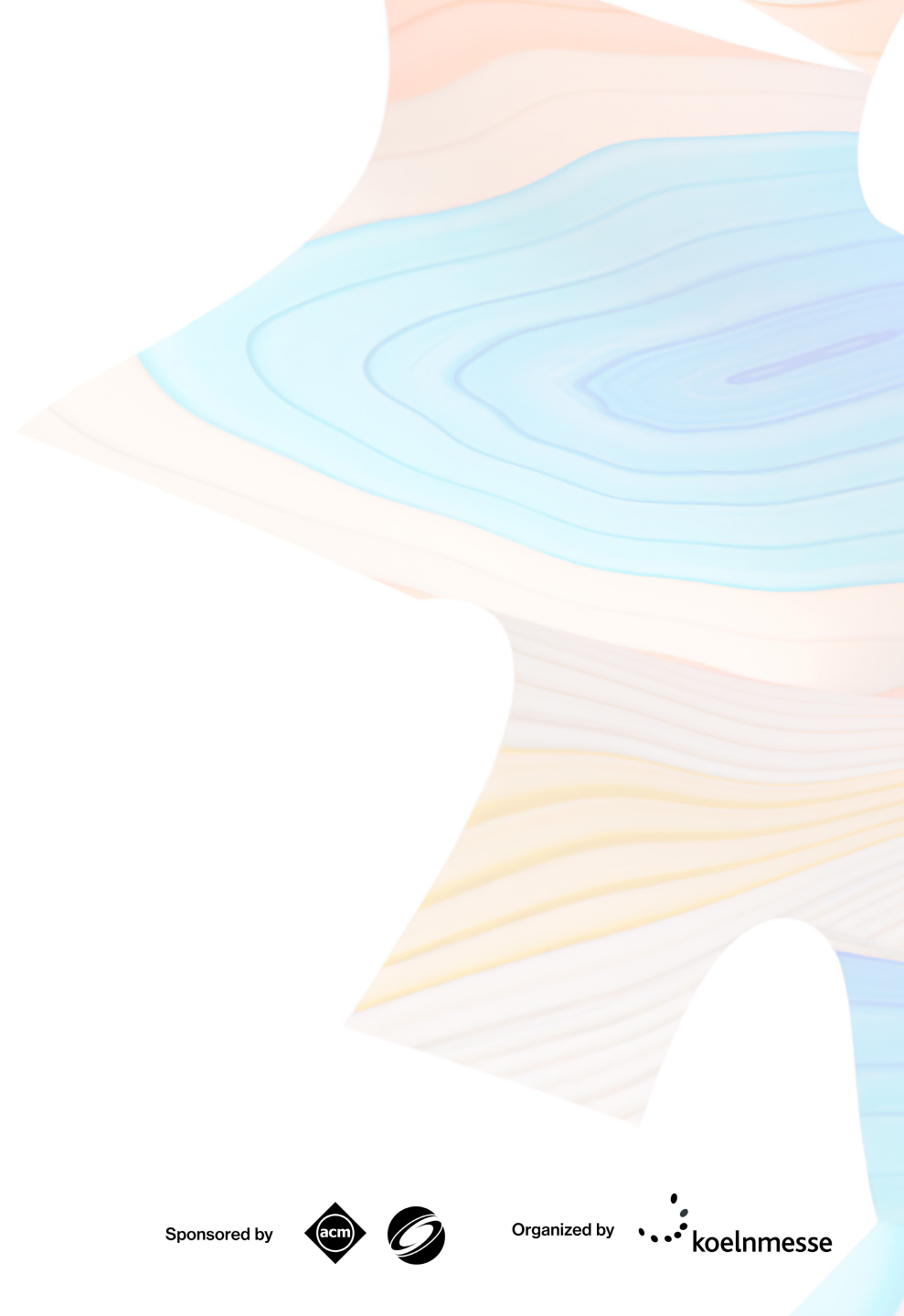
# stochastic gradients

target PDE solution

initial PDE solution



min  max

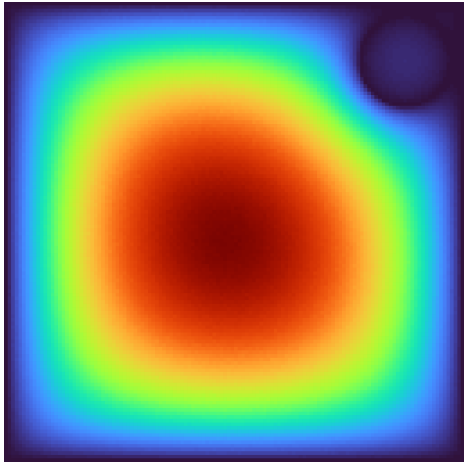




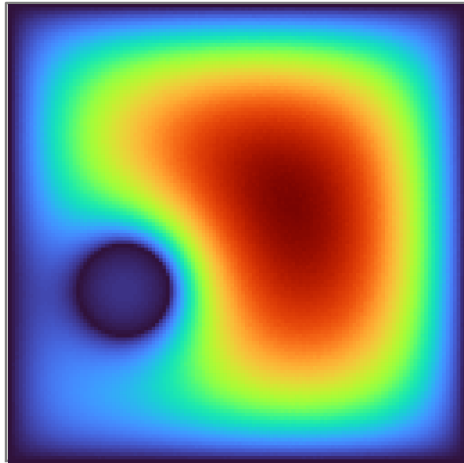
# stochastic gradients

optimization trajectory at **equal number of iterations**

target PDE solution



initial PDE solution

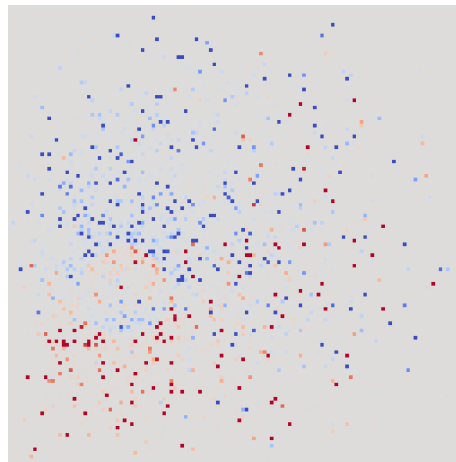
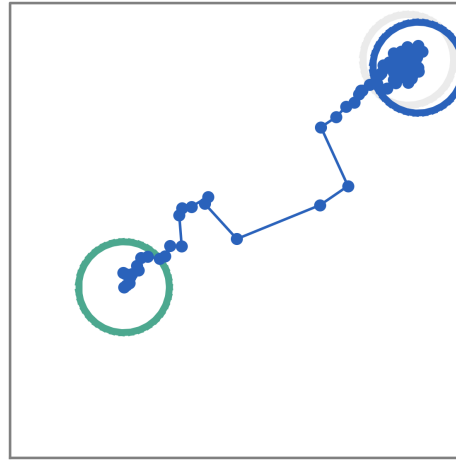


min  max

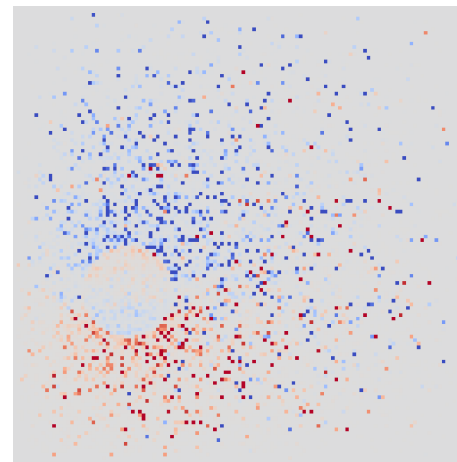
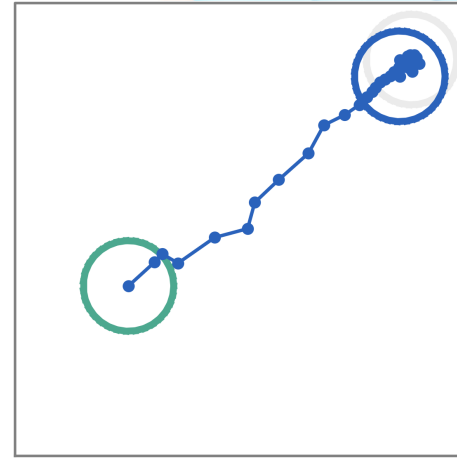
y-coordinate derivative  
visualized on first iteration

min  max

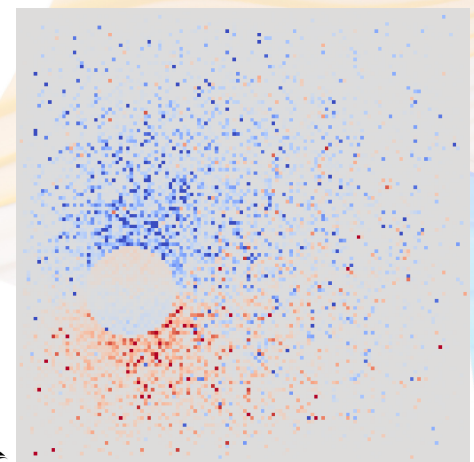
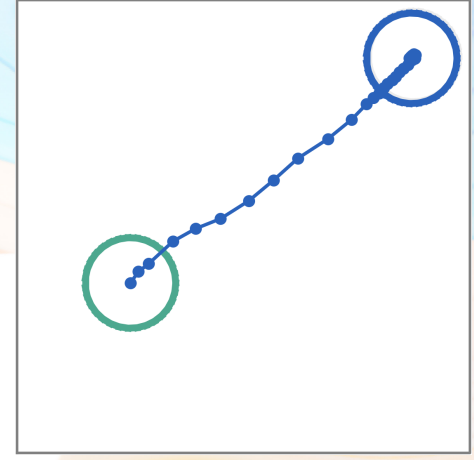
2 walks per point



8 walks per point



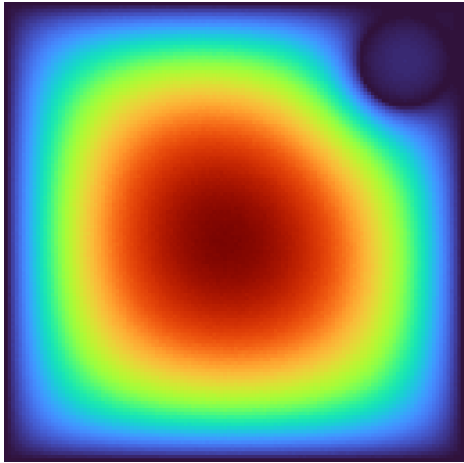
32 walks per point



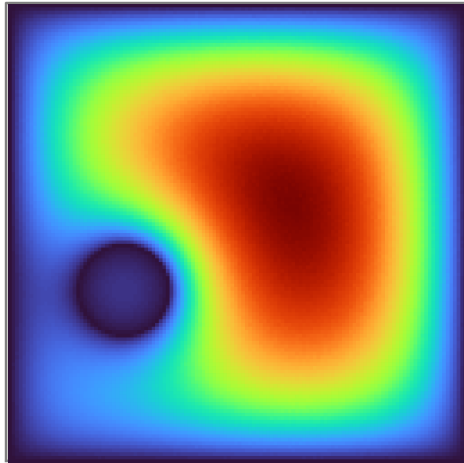
# stochastic gradients

optimization trajectory at **equal time**

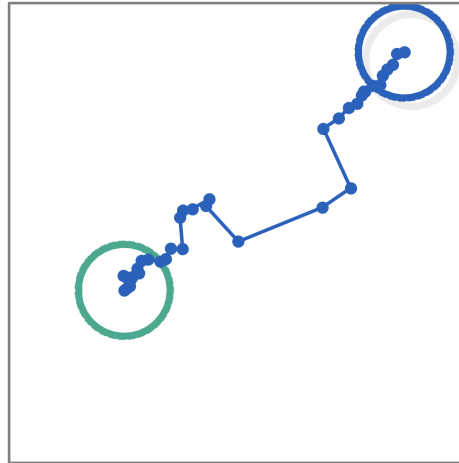
target PDE solution



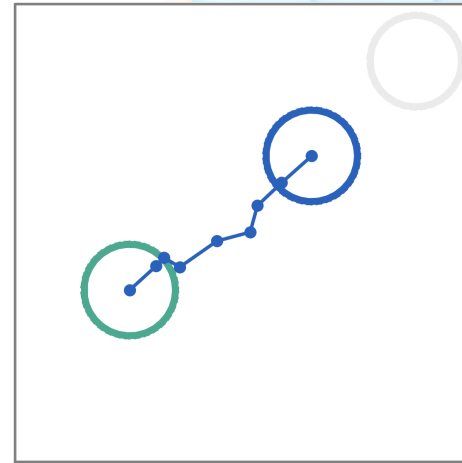
initial PDE solution



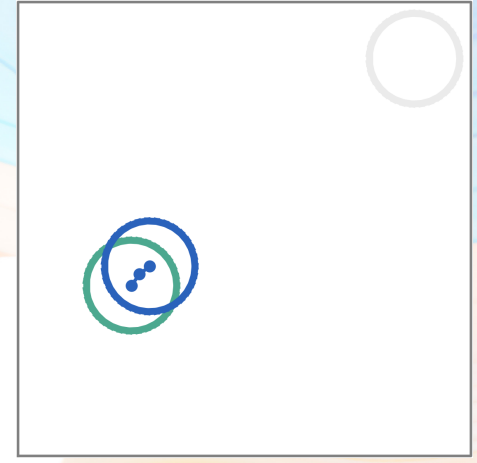
2 walks per point



8 walks per point



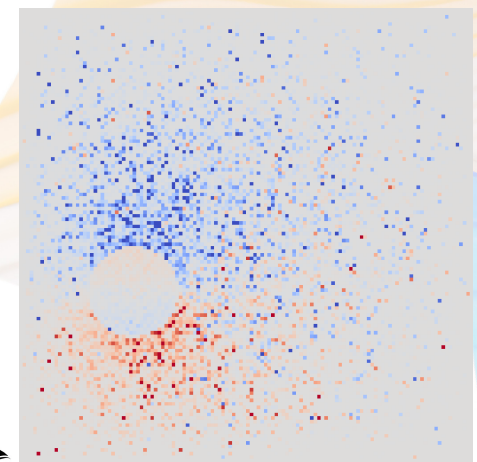
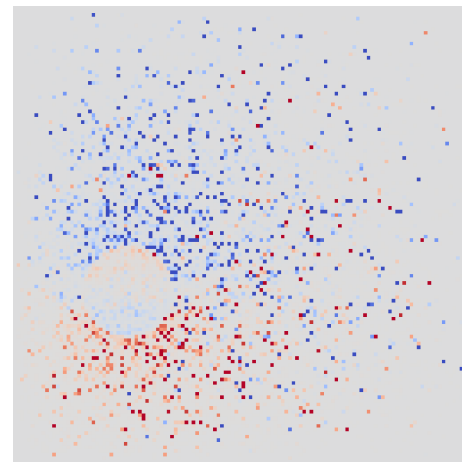
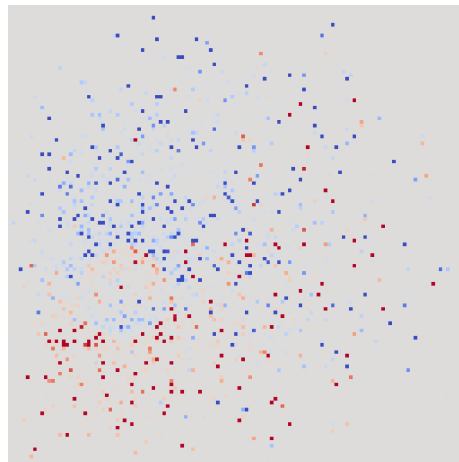
32 walks per point



min  max

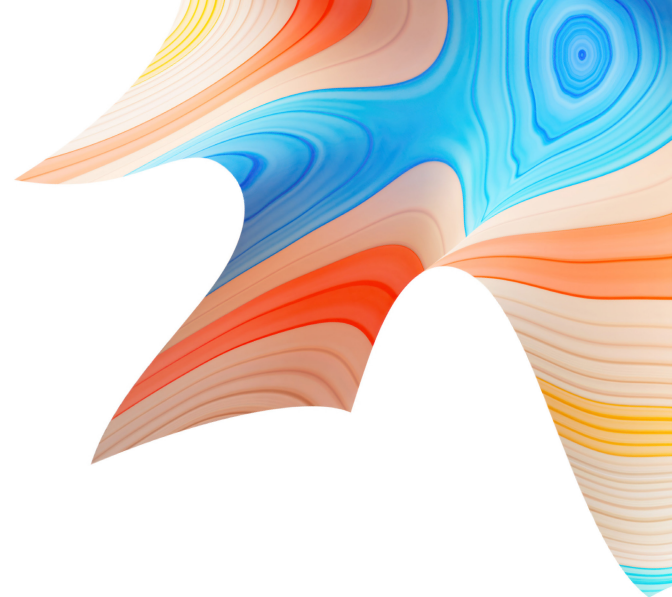
Noisy derivatives can improve performance!

max

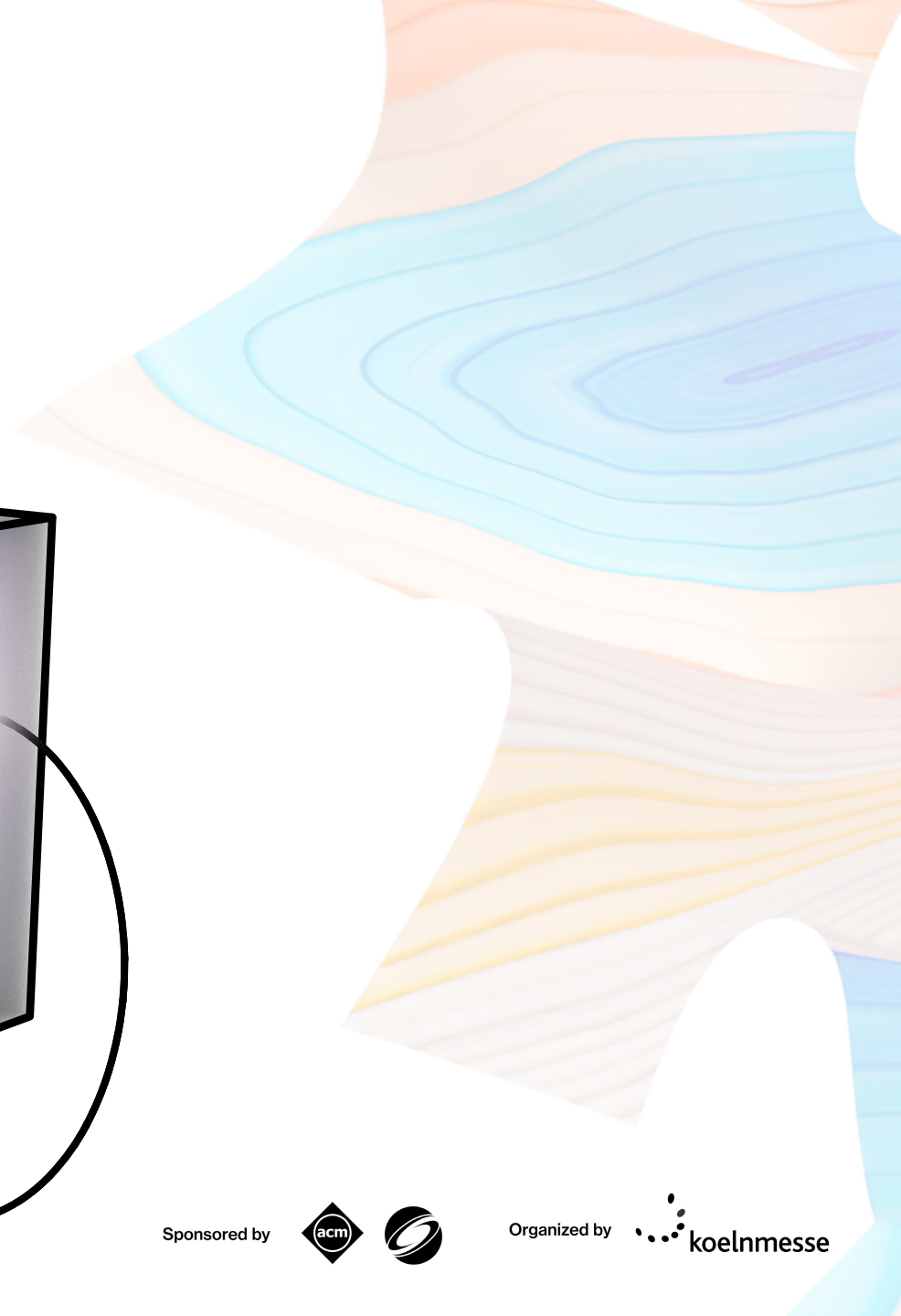
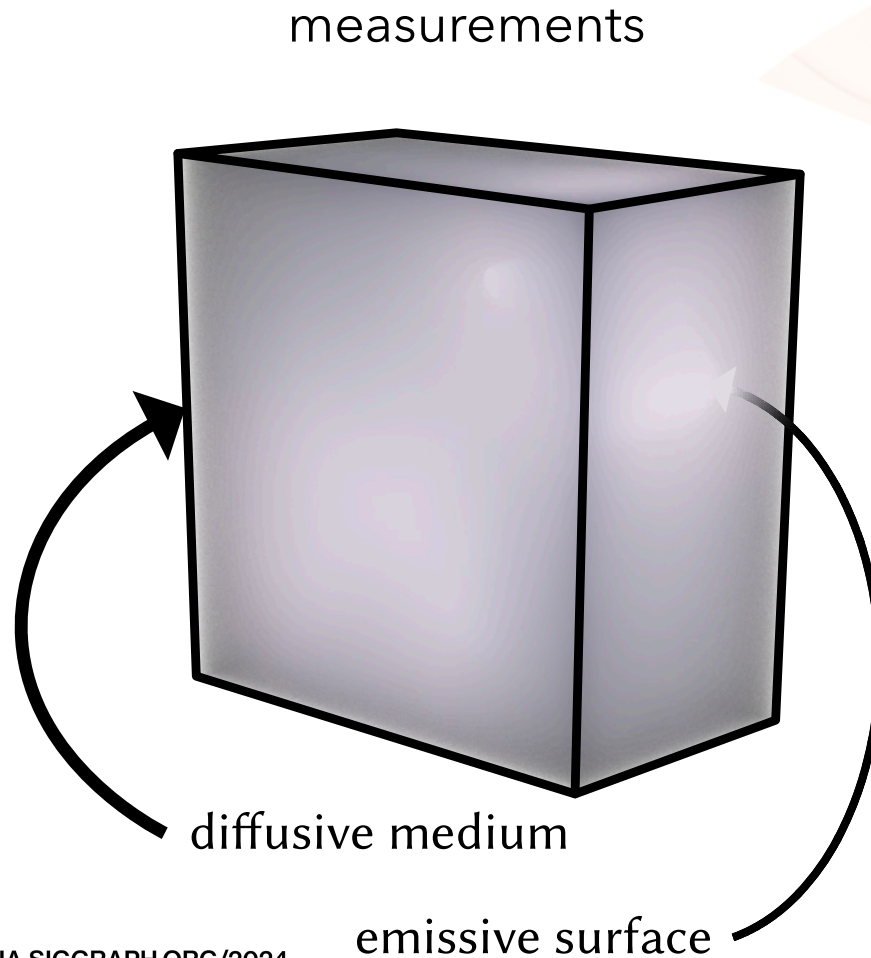




# applications

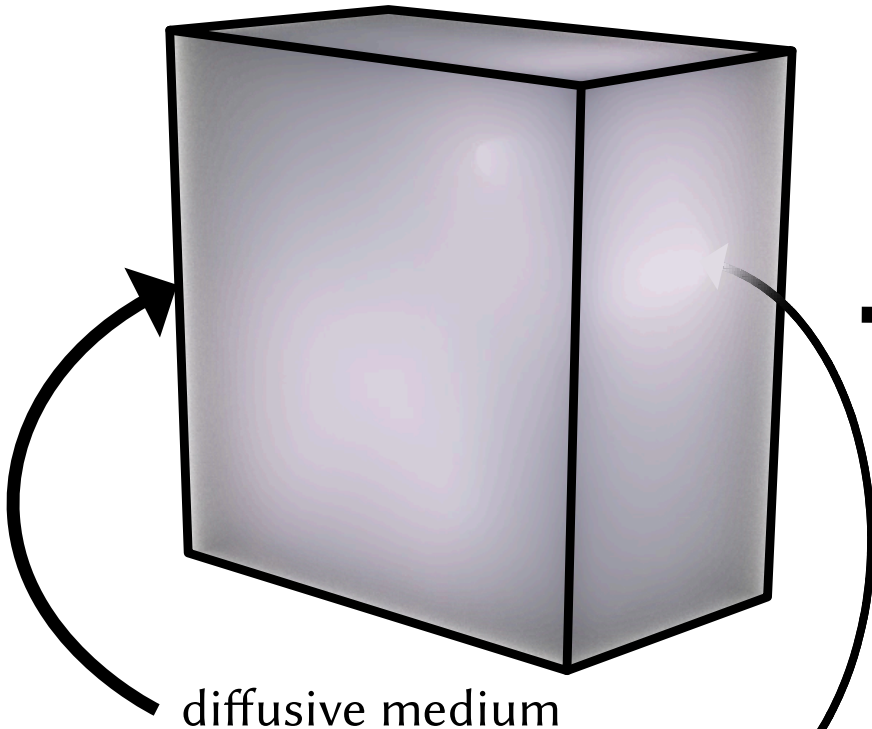


# Shape from diffusion

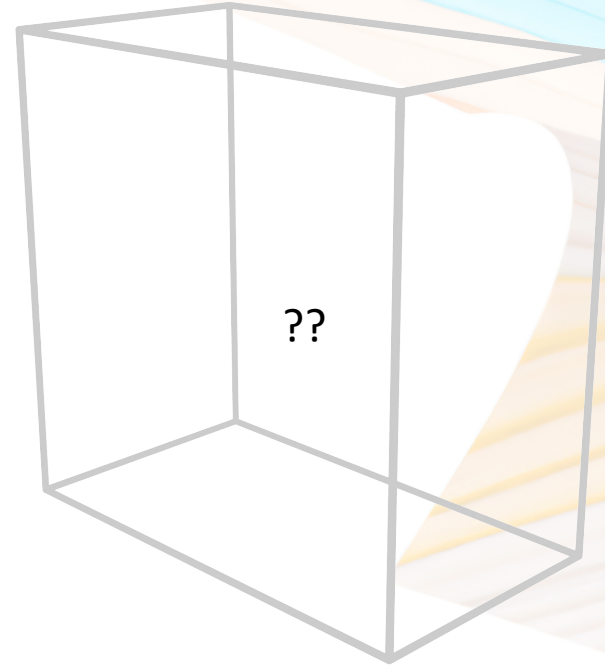


# Shape from diffusion

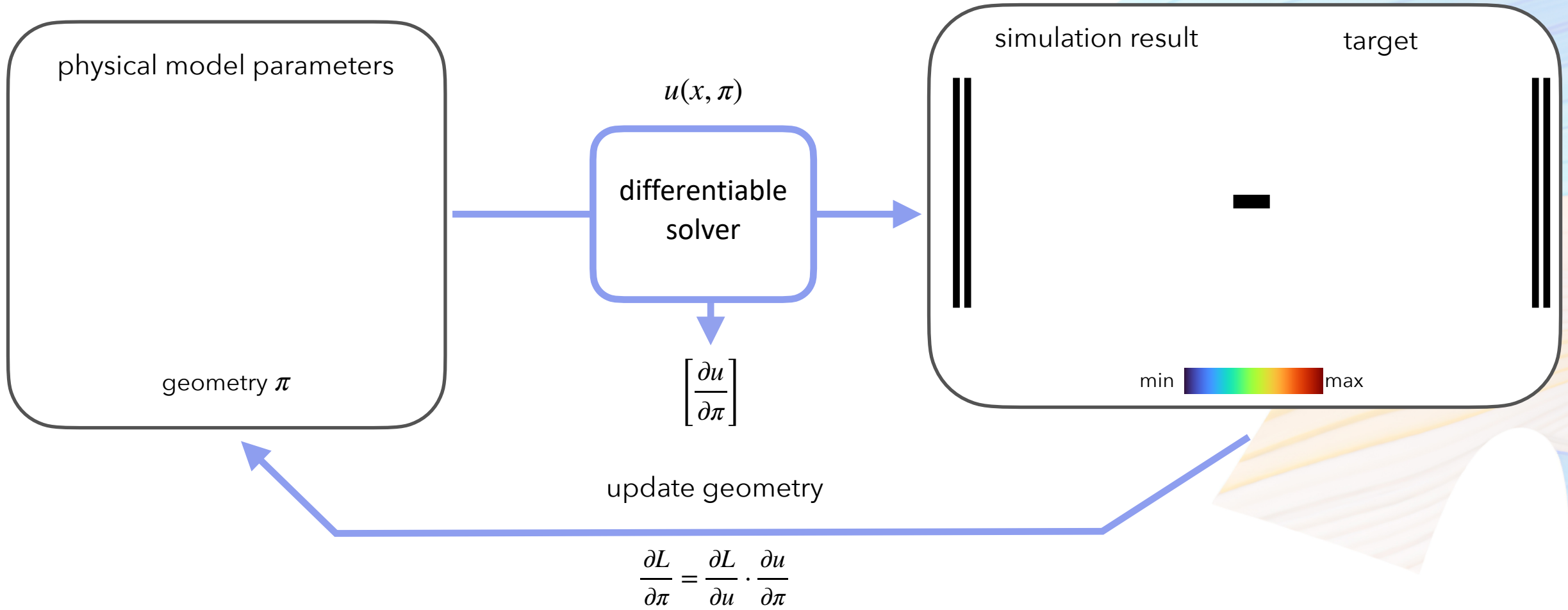
measurements



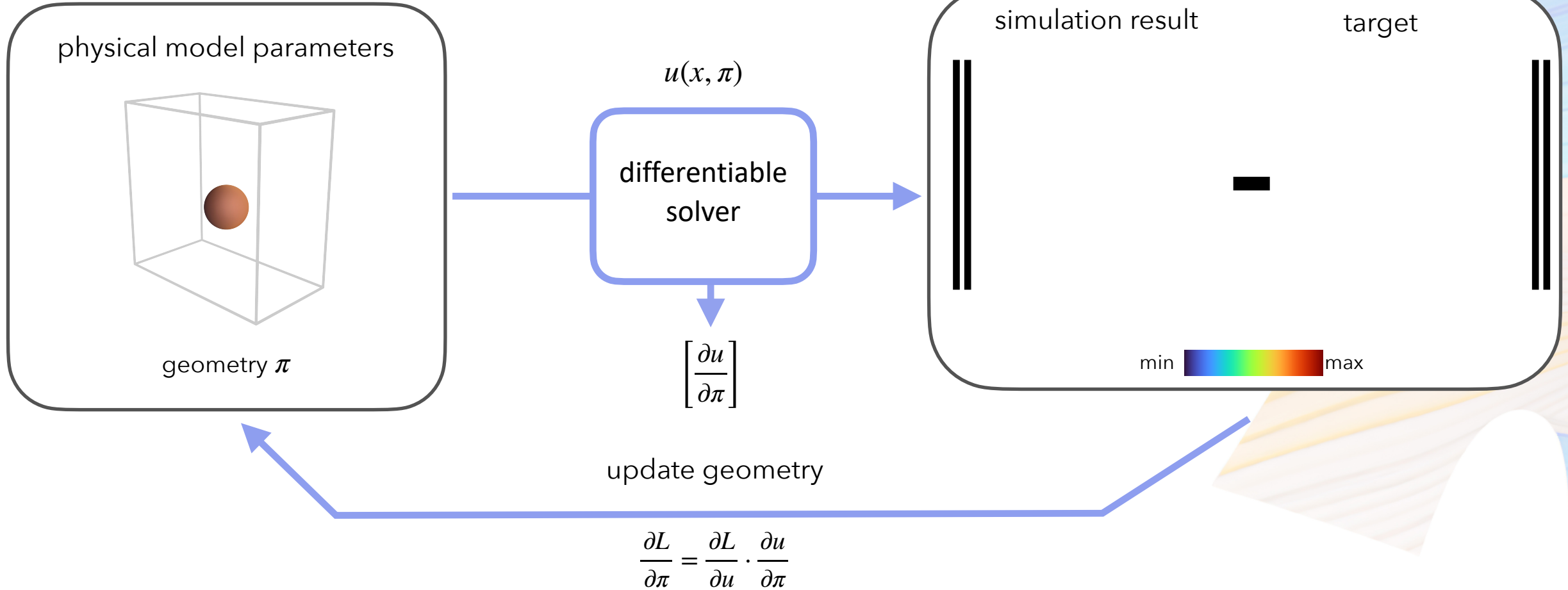
shape of emissive surface



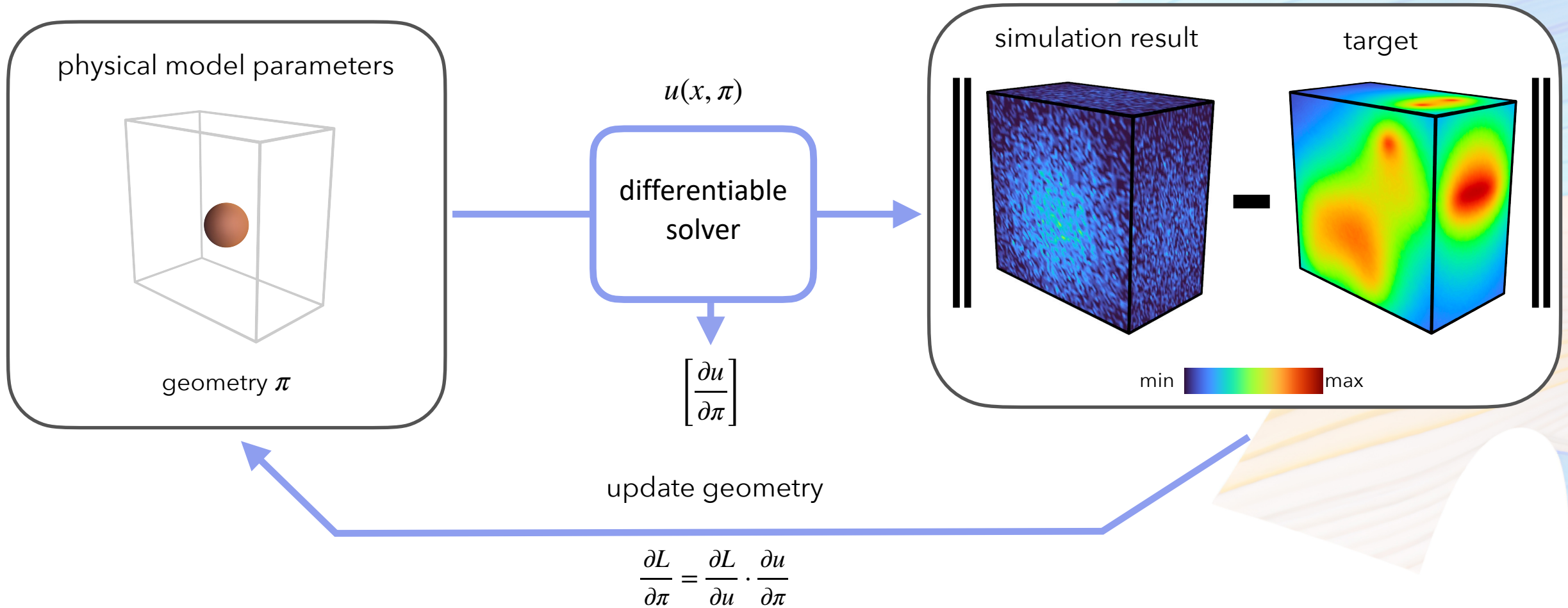
# Shape from diffusion



# Shape from diffusion

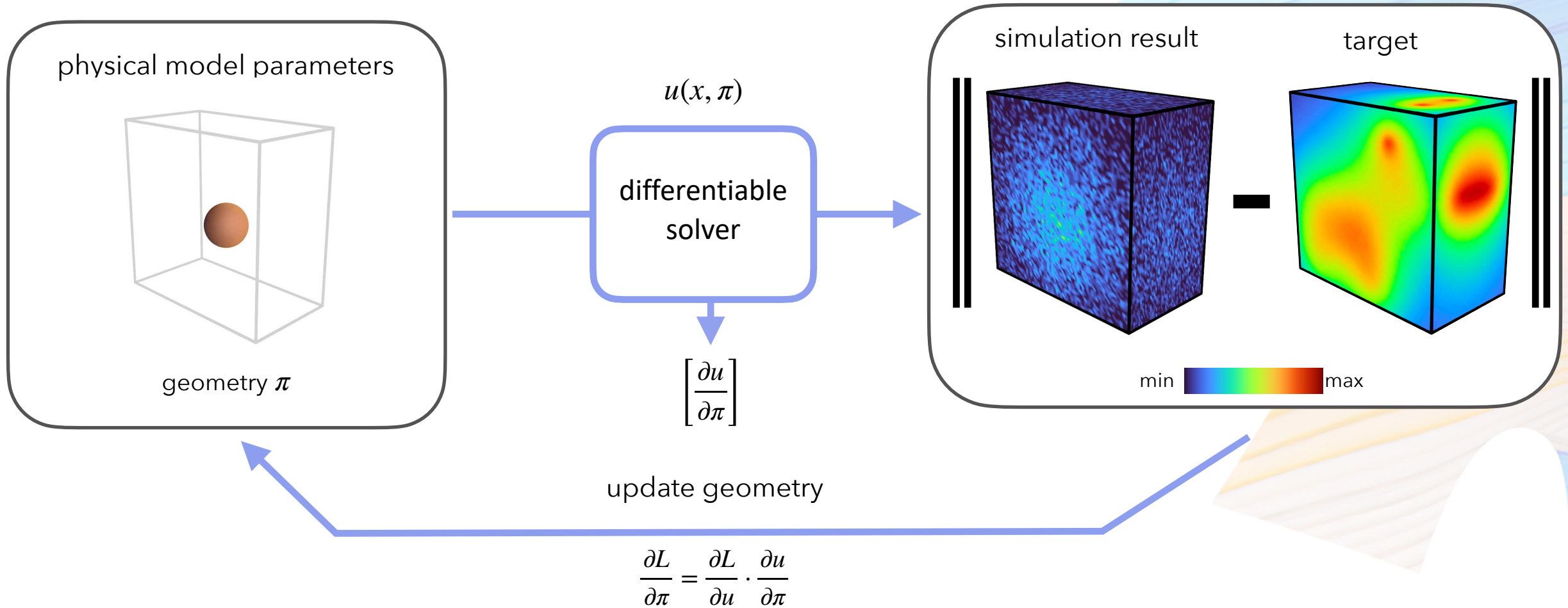


# Shape from diffusion

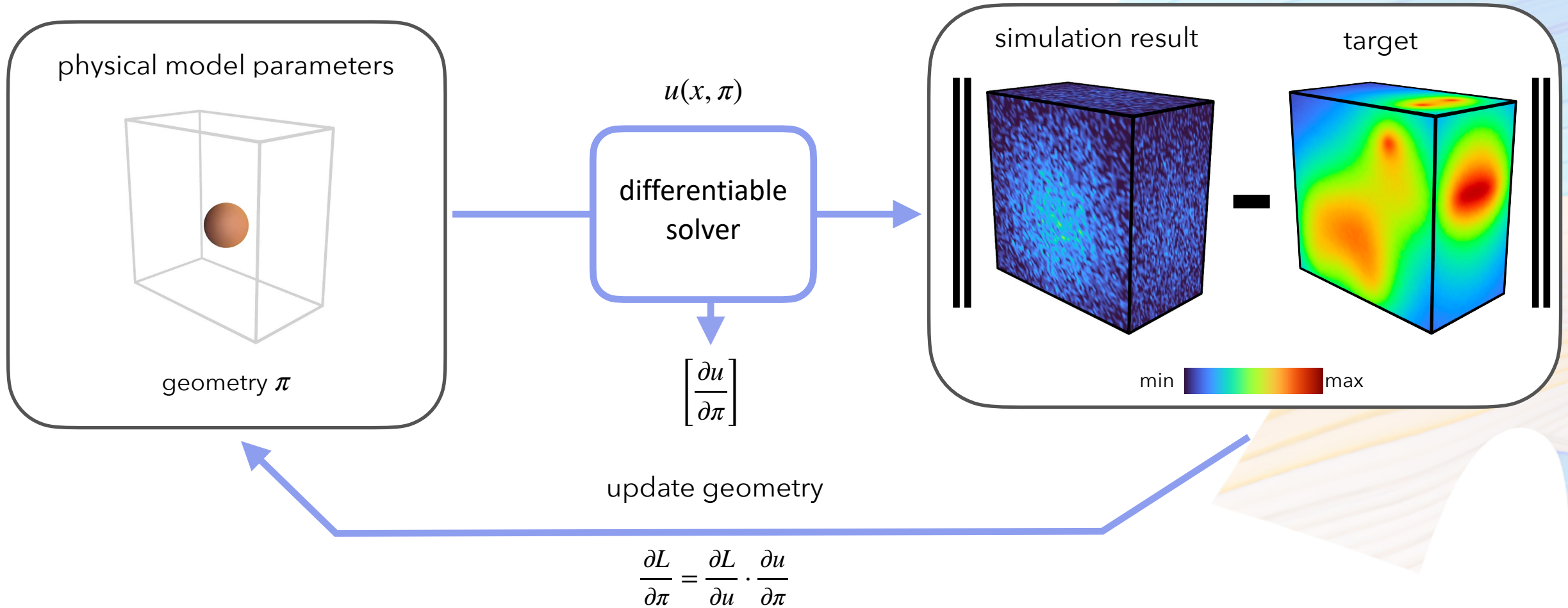




# Shape from diffusion

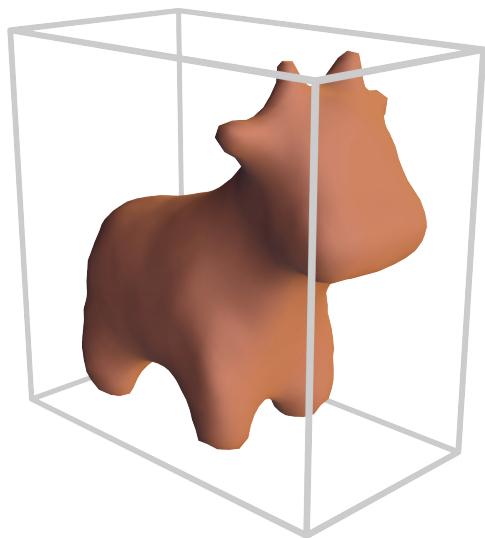


# Shape from diffusion

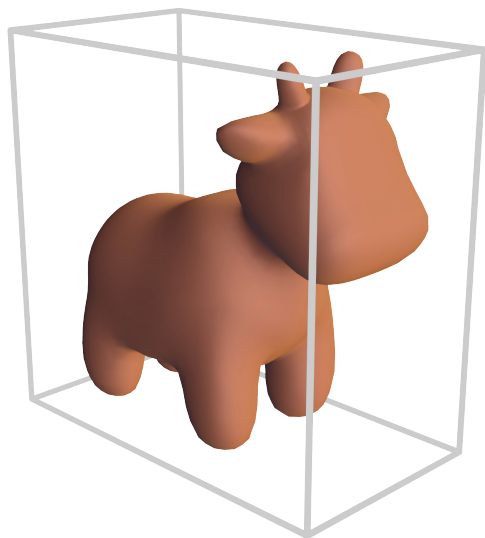


# Shape from diffusion results

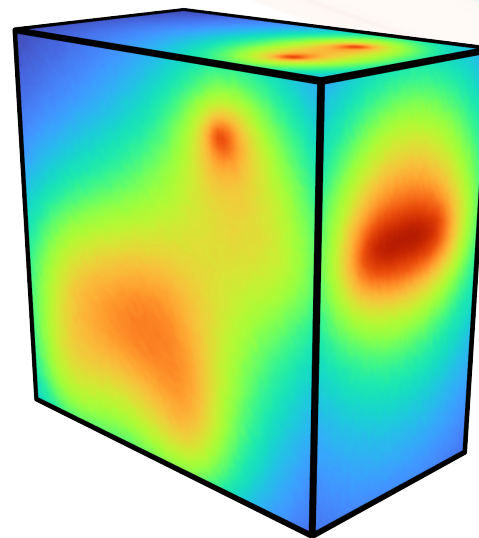
geometry



optimized

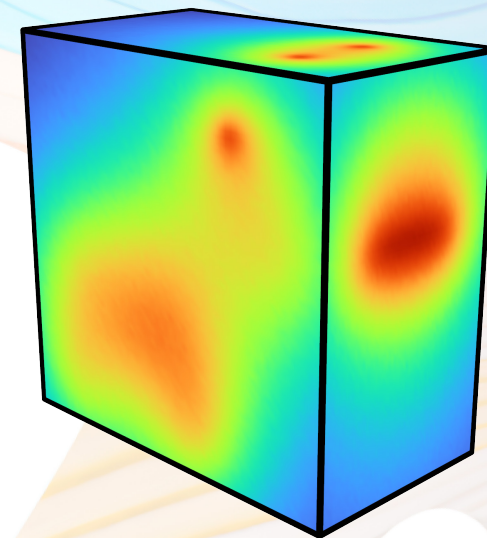


reference



optimized

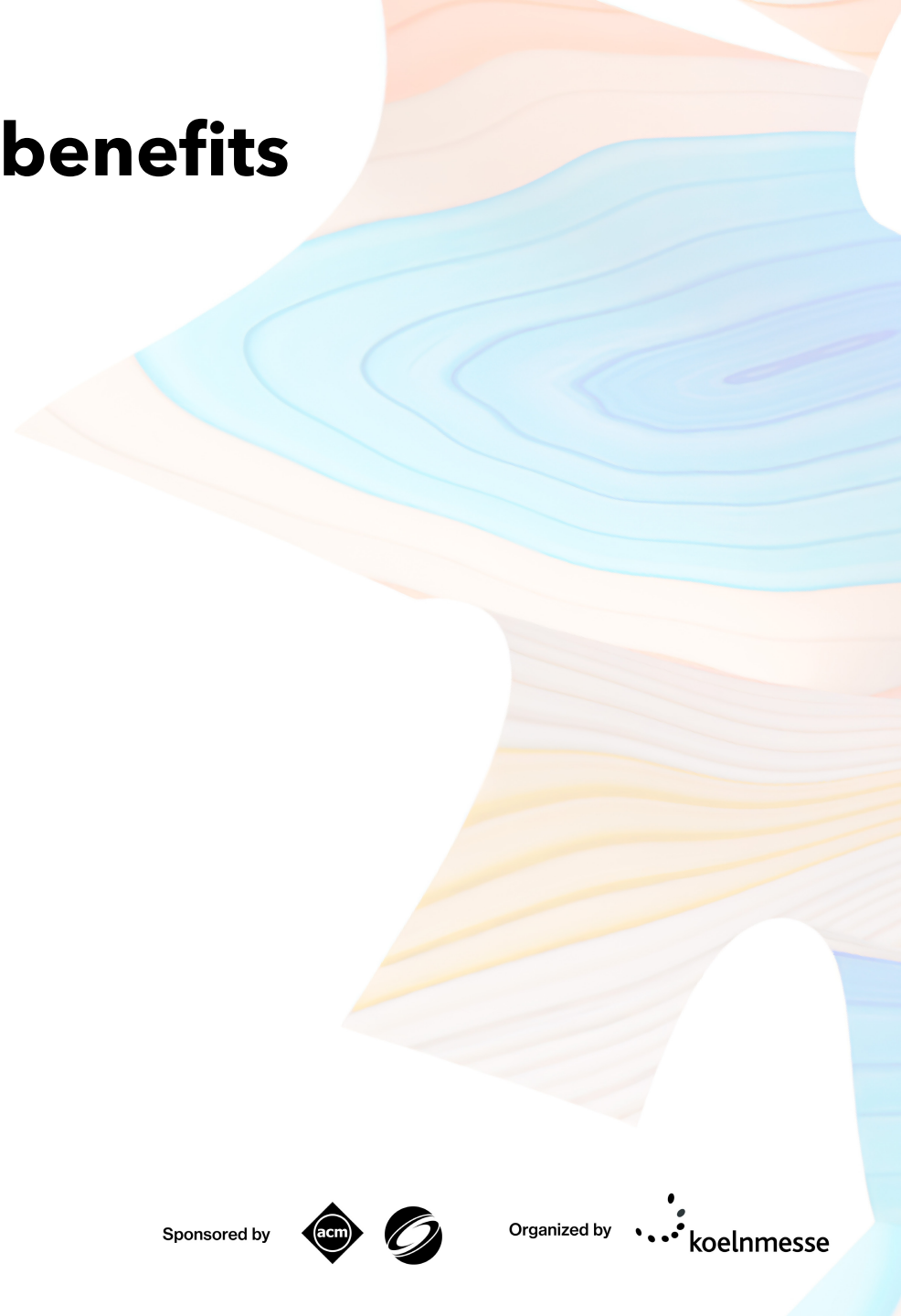
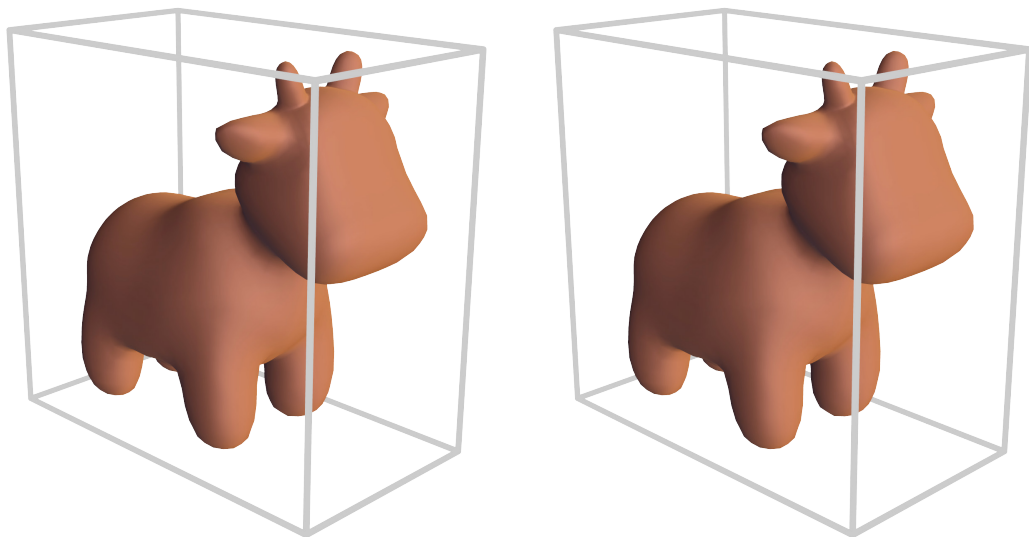
solution



target

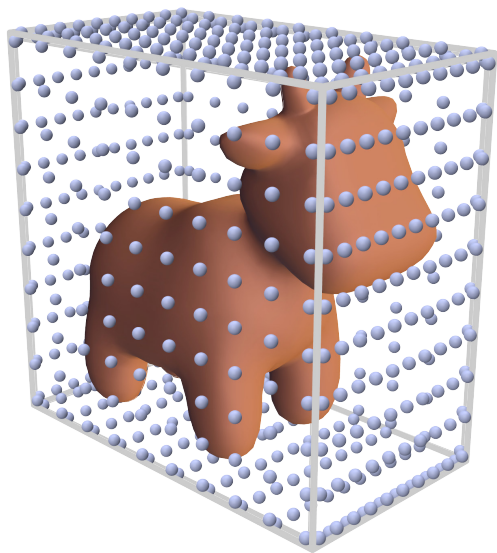
# Shape from diffusion benefits

pointwise evaluation

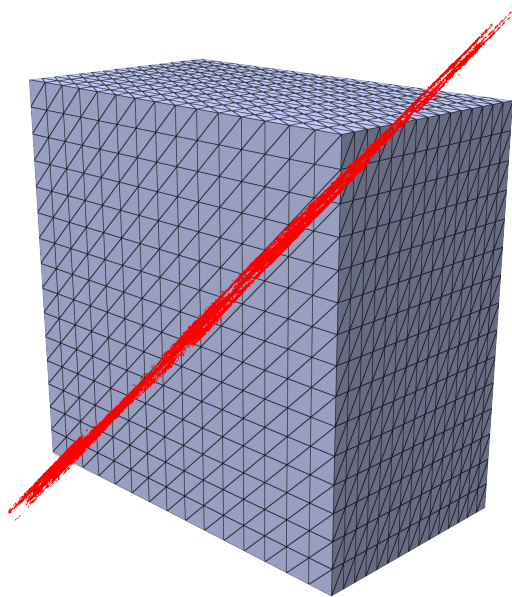


# Shape from diffusion benefits

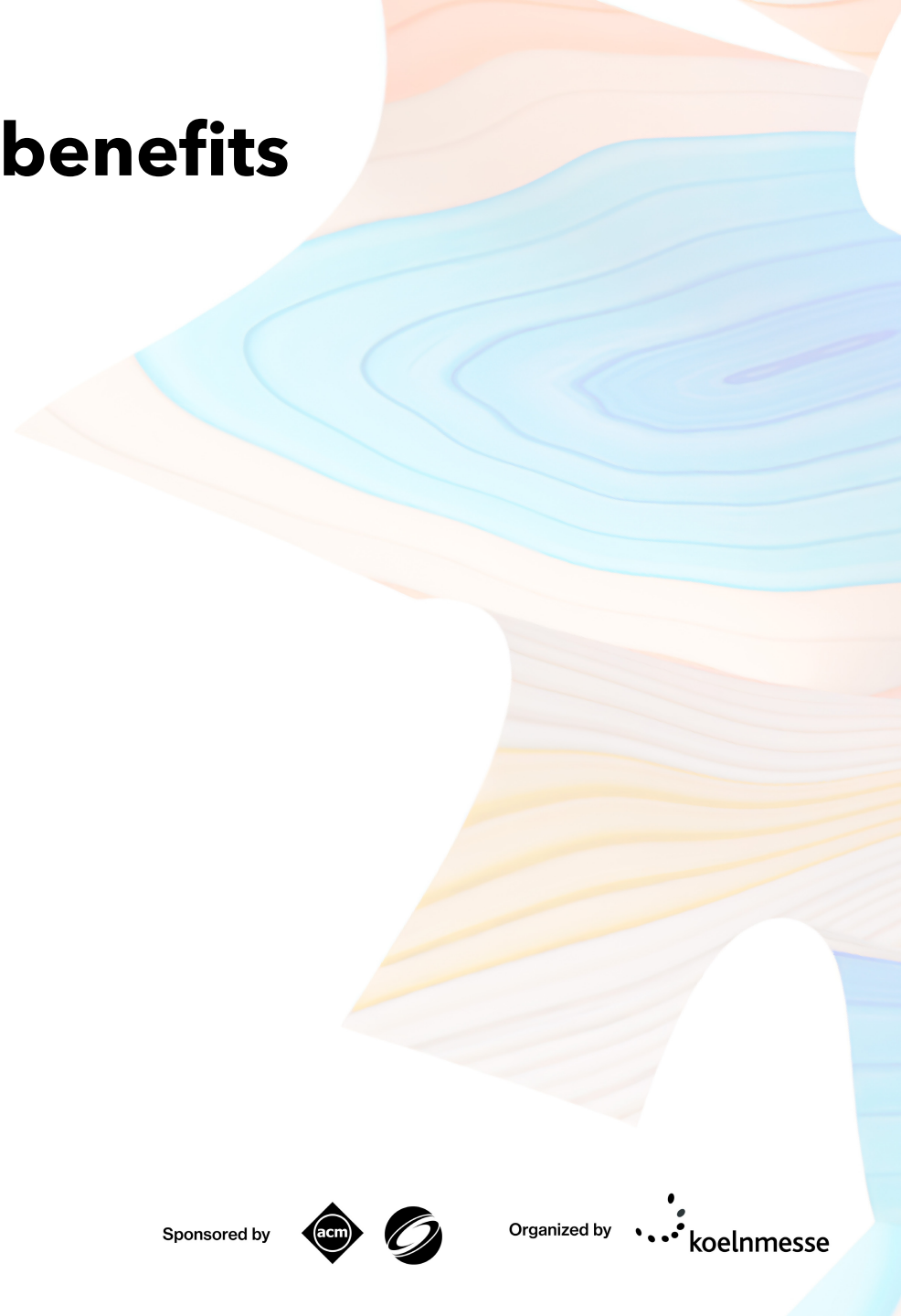
pointwise evaluation



surface points



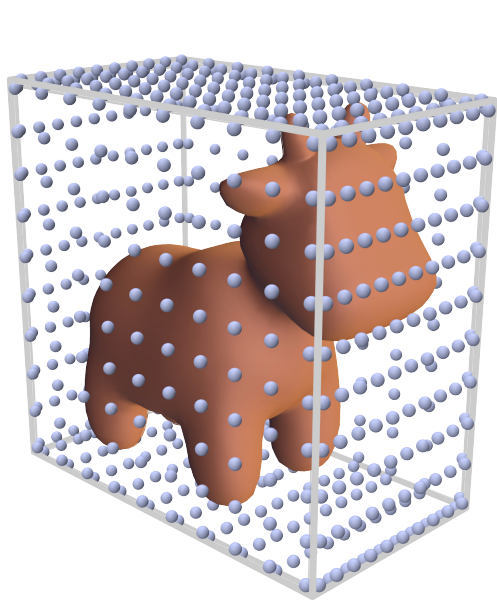
dense volumetric grid



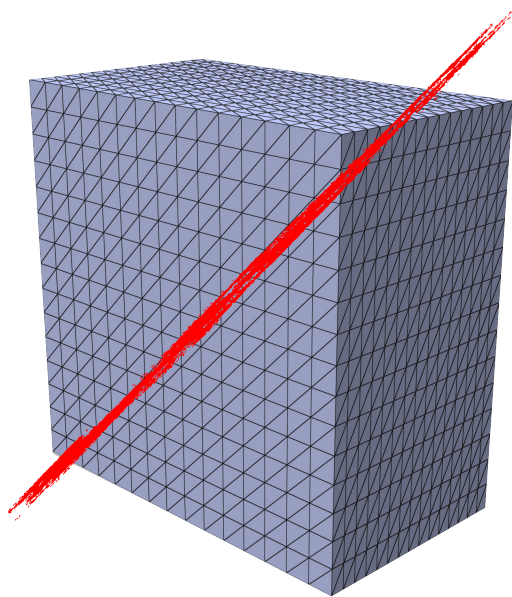
# Shape from diffusion benefits

pointwise evaluation

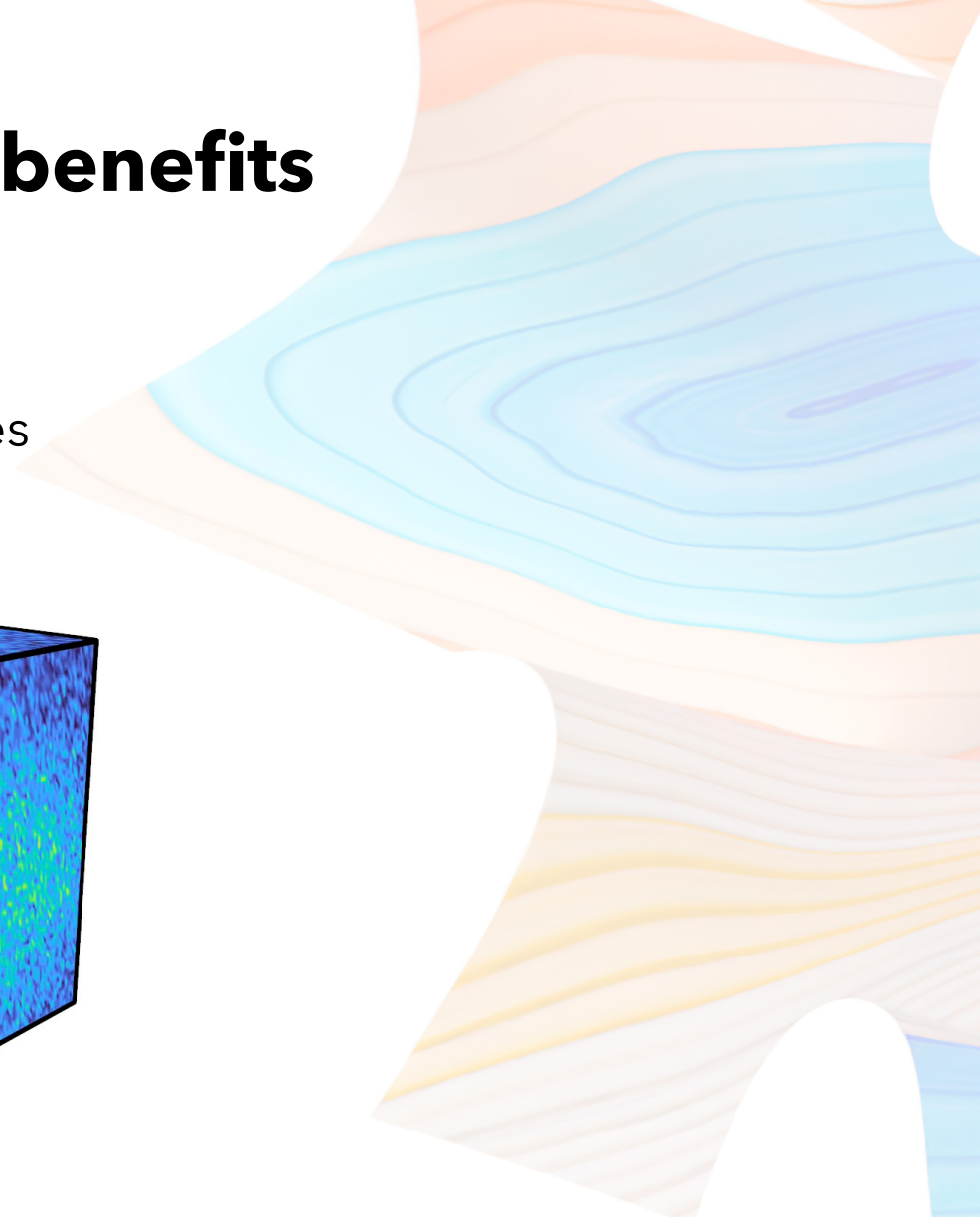
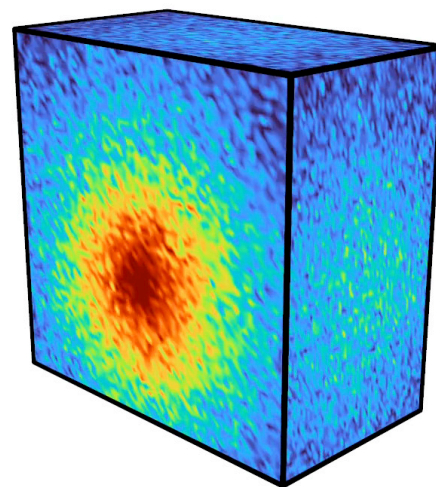
noisy estimates



surface points



dense volumetric grid

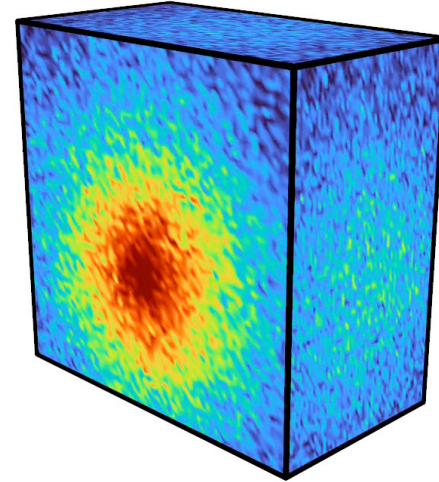
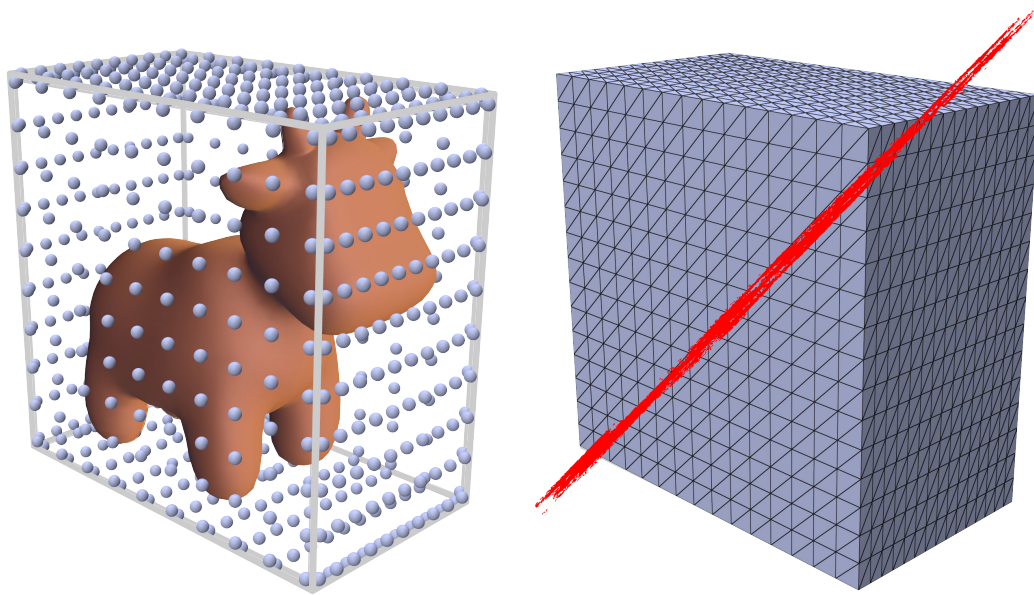


# Shape from diffusion benefits

pointwise evaluation

noisy estimates

borrow ideas from rendering



surface points

dense volumetric grid



**Large Steps in Inverse Rendering of Geometry**

BAPTISTE NICOLET, *École Polytechnique Fédérale de Lausanne (EPFL), Switzerland*  
ALEC JACOBSON, *University of Toronto, Canada*  
WENZEL JAKOB, *École Polytechnique Fédérale de Lausanne (EPFL), Switzerland*

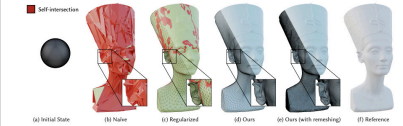


Fig. 1. An inverse reconstruction of the Neumann bear from spherical starting points with 20 rendered views (1) shows (2) Noise application of a differentiable renderer produces an available target mesh when gradient steps fall on the silhouette without regard for distortion or self-intersections. (3) Regularization can alleviate such problems by making the optimization aware of mesh quality. On the flipside, this procedure uses smooth parts of the geometry and causes unsatisfactory convergence to gradient-based optimizers. While the final mesh undergoes further better, a clear inspection of the silhouette rendering reveals noticeable self-intersections. (4) Our method addresses both problems and converges to a high-quality mesh. (5) Combined with an image rendering step, our reconstruction captures fine details of the reference (5). The hyperparameters of each method were optimized to obtain the best convergence at equal time. Self-intersections are shown in red.

Inverse reconstruction from images is a central problem in many scientific and engineering disciplines. Recent progress in differentiable rendering has led to methods that can efficiently differentiate the full process of image formation with respect to millions of parameters to solve such problems via gradient-based optimization.

At the same time, the availability of cheap derivatives does not necessarily make an inverse problem easy to solve. Grid-based representations remain a practical source of artifacts, as sparse gradient step landing on vertices produces small gaps in the mesh. Implicit methods avoid such artifacts, but all-in-one optimization, or lead to inadequate usage of the entire budget due to distortion. These types of issues are often intractable in the sense that subsequent optimization steps will further exacerbate them. In other words, the optimization landscape is often non-convex.

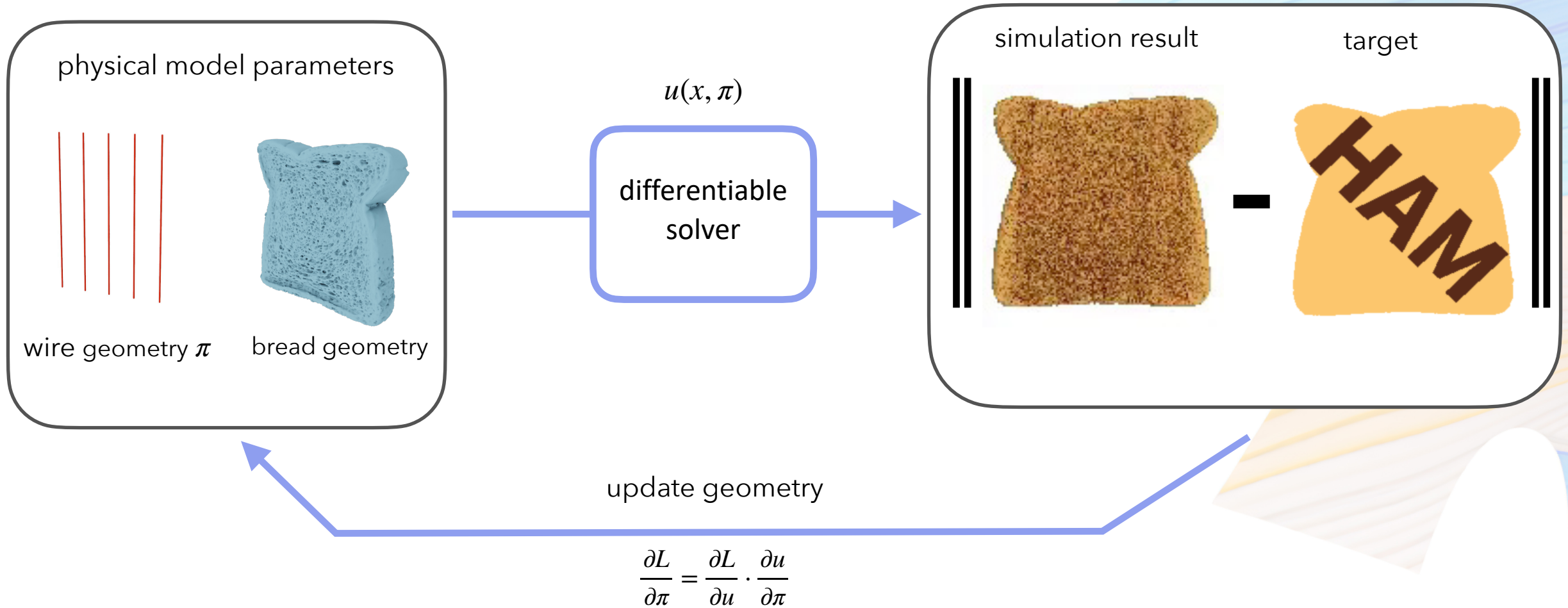
Such limitations arise and are commonly mitigated by imposing additional regularization, typically in the form of Laplacian energies that quantify and regularize the smoothness of the mesh. However, regularization introduces its own set of problems, solutions must now compromise between solving the problem and being smooth. Furthermore, gradient steps involving vertices' neighbors becomes noisy, and the optimization process becomes more complex.

Authors' addresses: Baptiste Nicolet, *École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland*; Alec Jacobson, *University of Toronto, Canada*; Wenzel Jakob, *École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland*.  
© 2021 Copyright held by the author(s). Published by the Association for Computer Graphics and Interactive Techniques, Inc. This is the author's version of the work, not the final published version. This is distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. ACM Reference Format: Nicolet, B., Jacobson, A., and Jakob, W. 2021. Large Steps in Inverse Rendering of Geometry. *ACM Trans. Graph.* 40, 4, Article 148 (December 2021), 13 pages. <https://doi.org/10.1145/3476133.3486161>

[Nicolet et al. 2021]

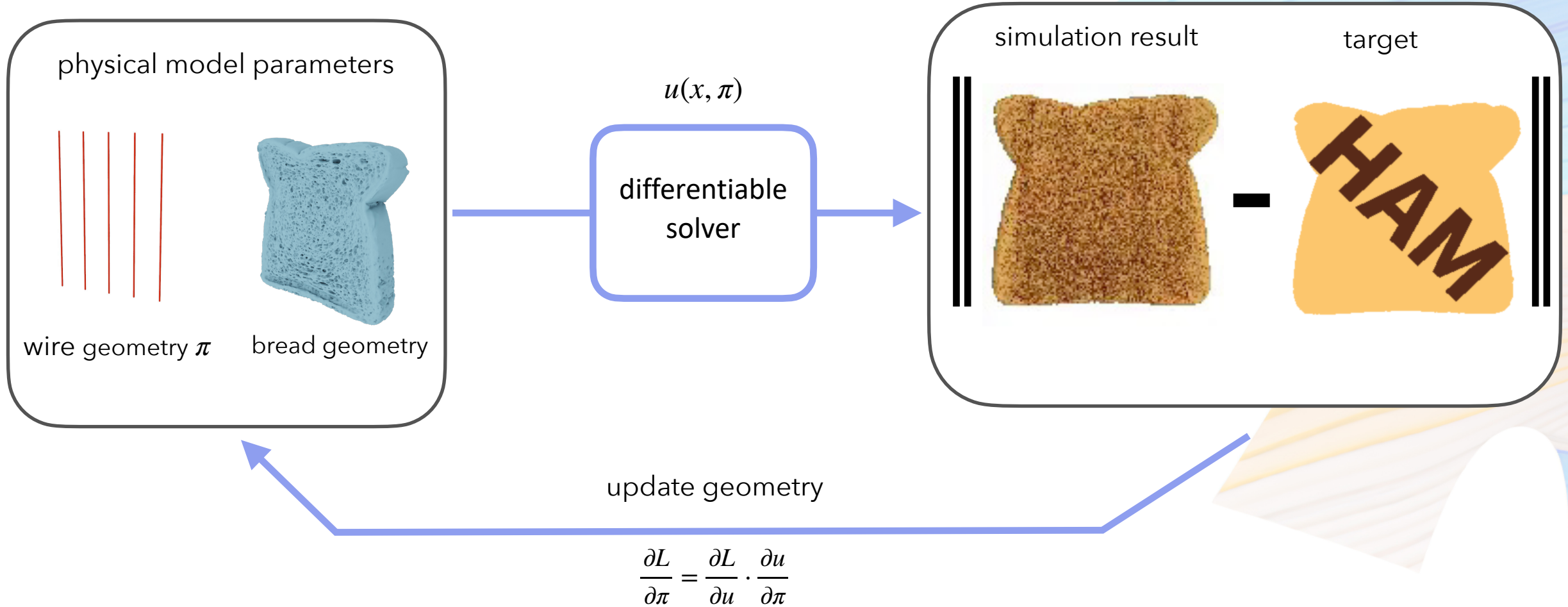


# Thermal design setup



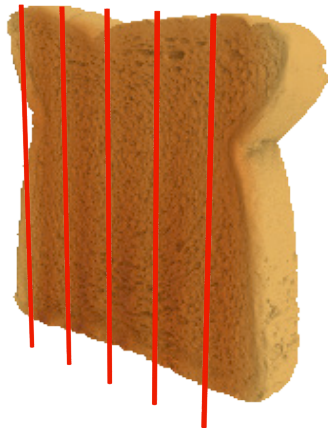
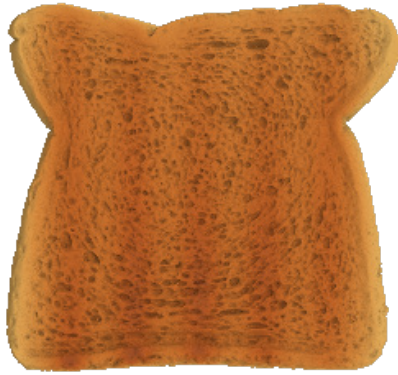


# Thermal design setup

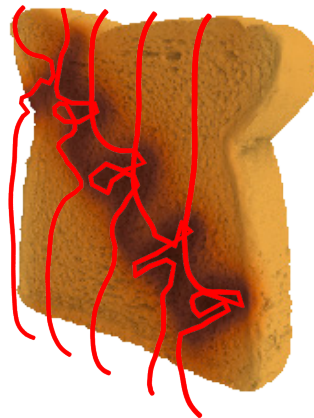
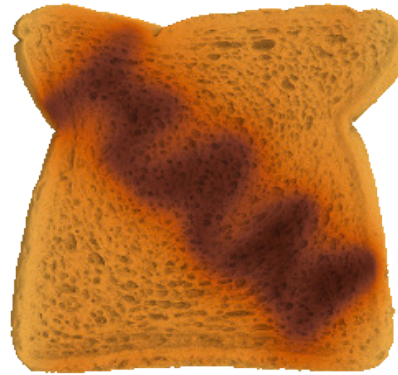


# Thermal design results

initial



optimized

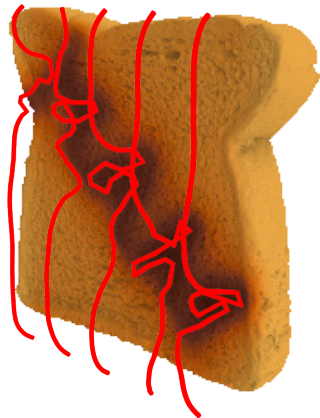
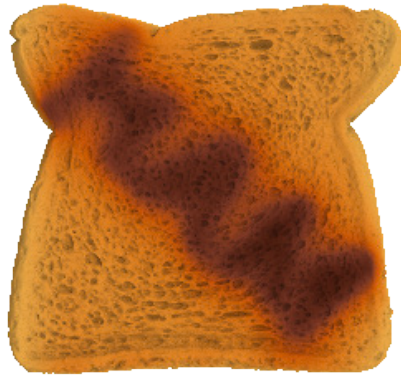


target



# Thermal design results

initial



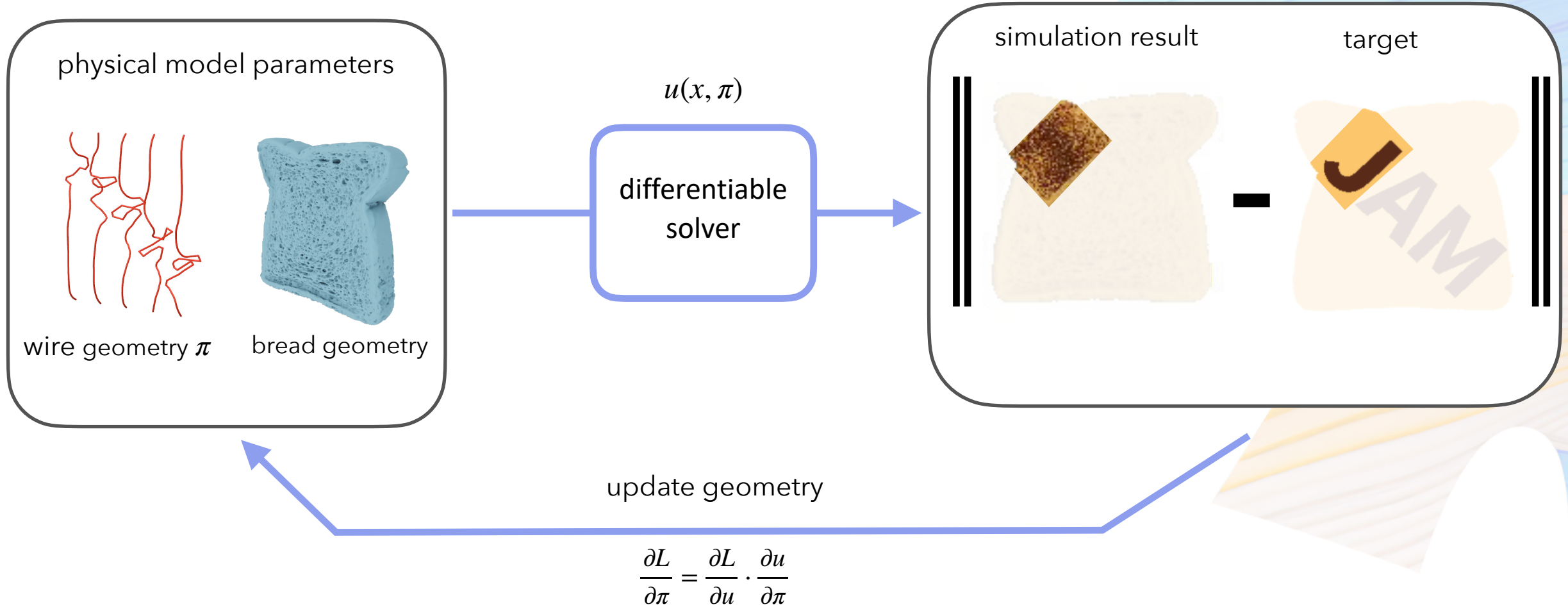
optimized

??

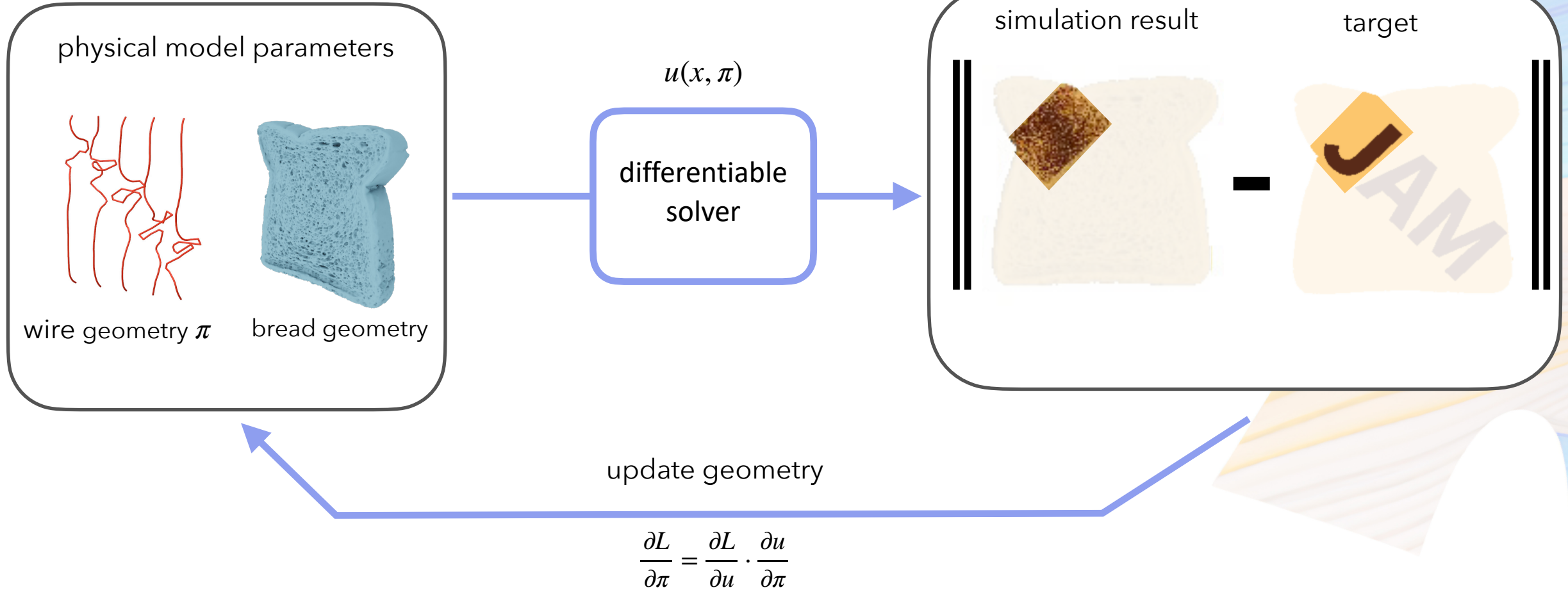
new target



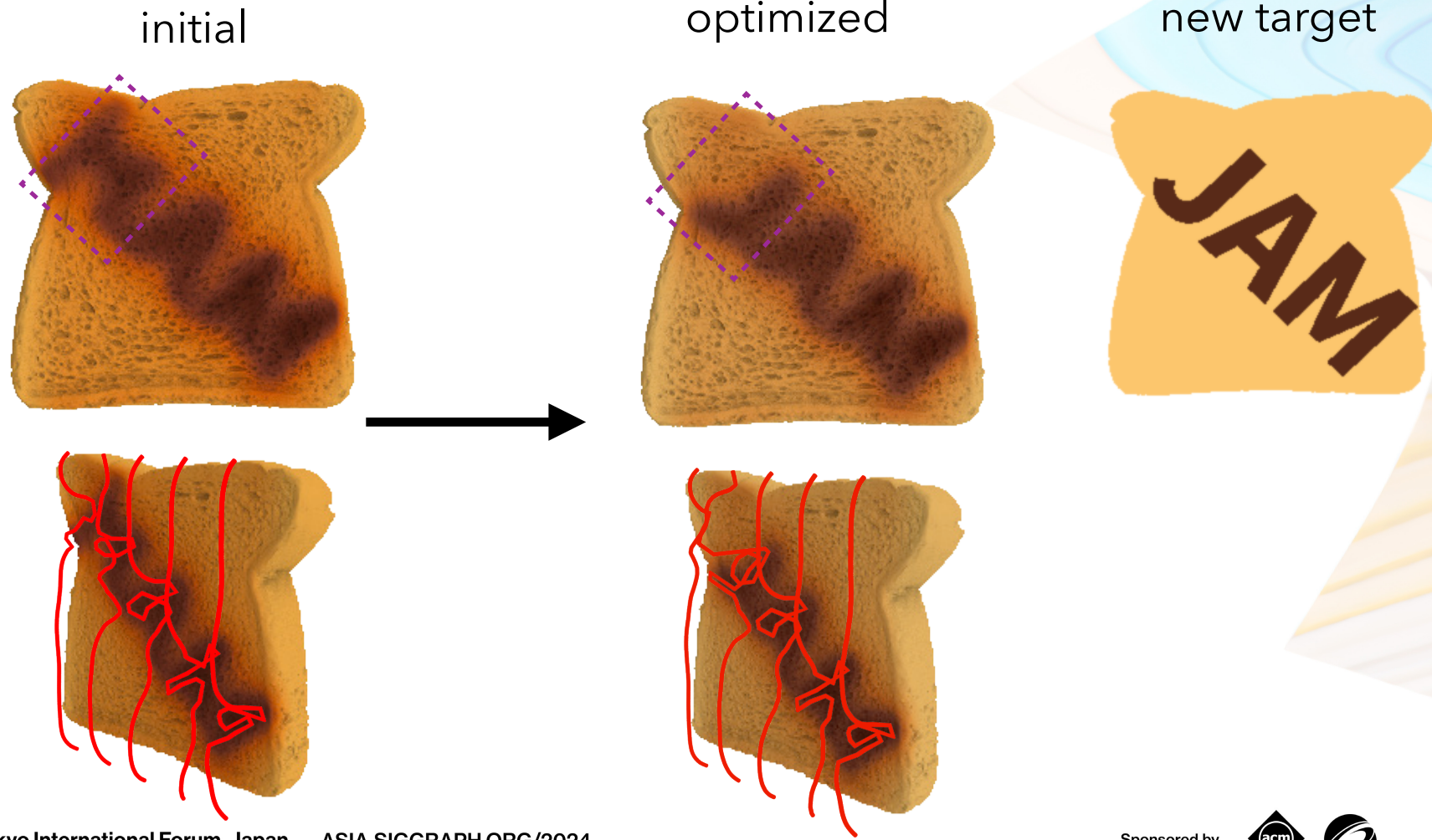
# Thermal design localized optimization



# Thermal design localized optimization

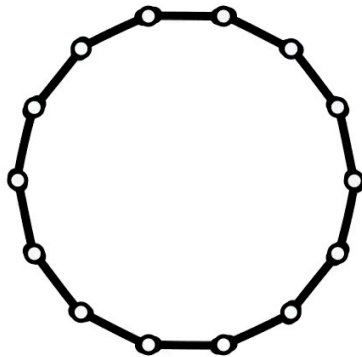


# Thermal design results



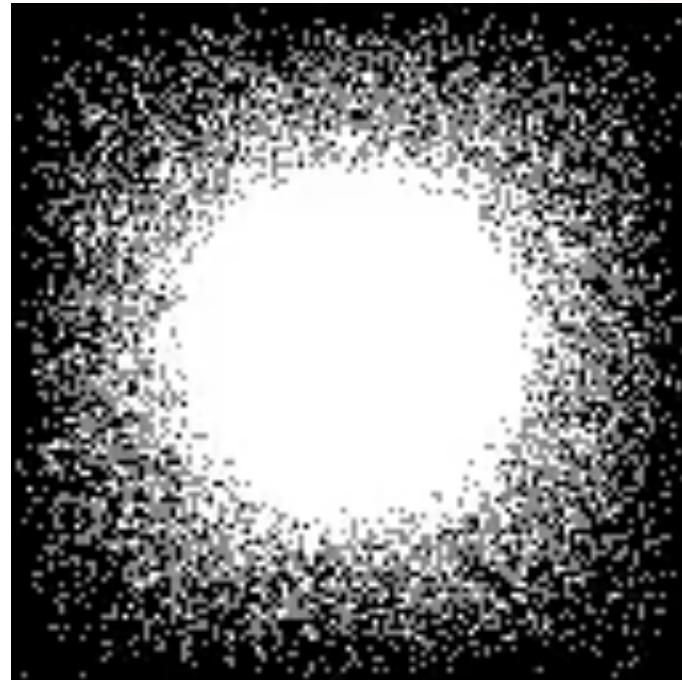
# Inverse diffusion curves

Bézier

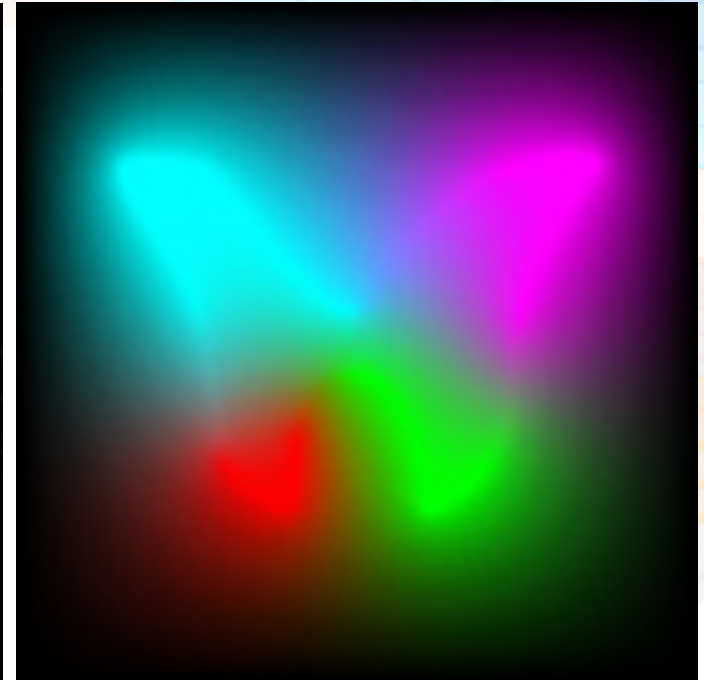


geometry

optimized



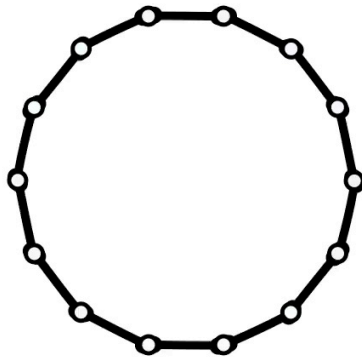
target



solution (RGB image)

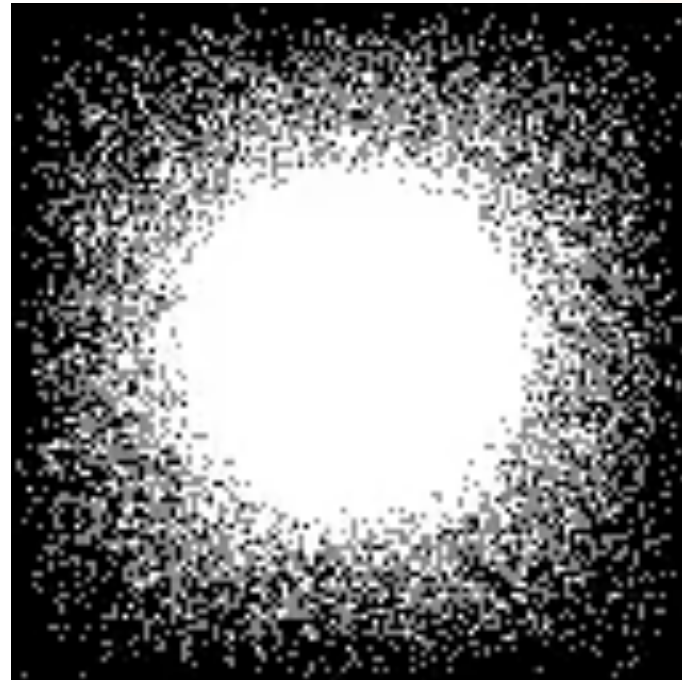
# Inverse diffusion curves

Bézier

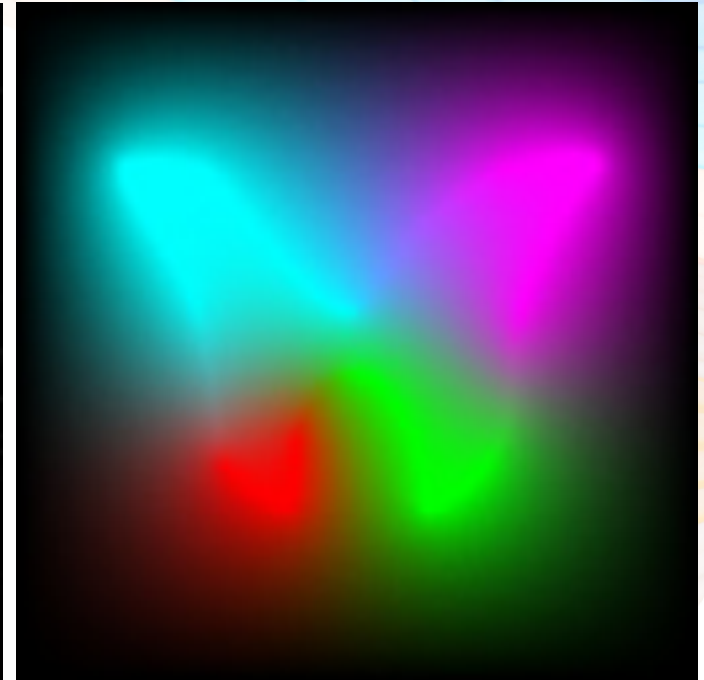


geometry

optimized



target

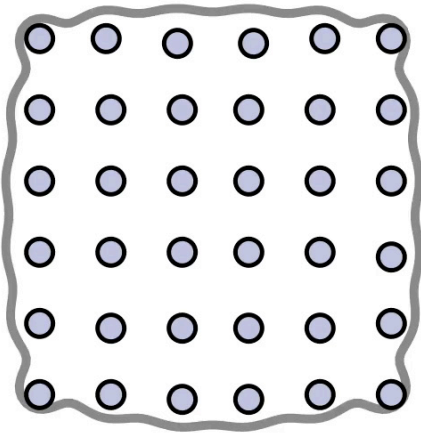


solution (RGB image)



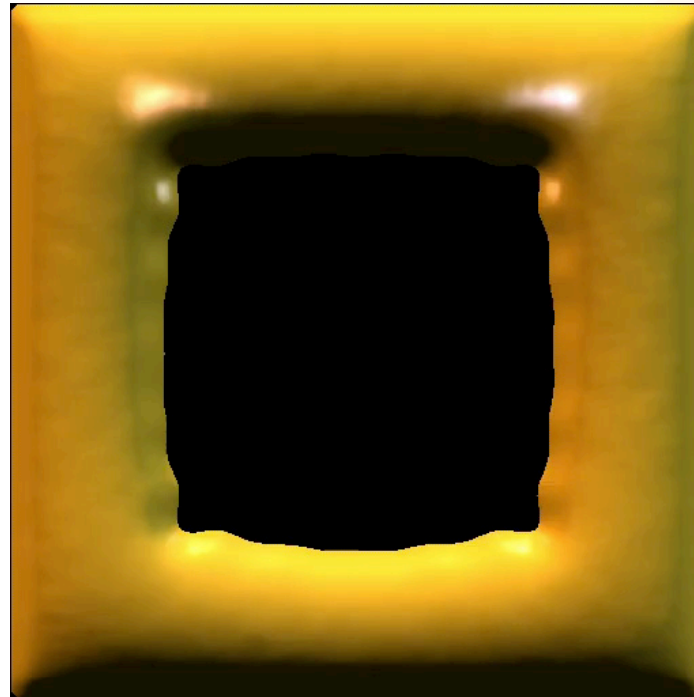
# Inflatable surfaces

implicit surface



geometry

optimized



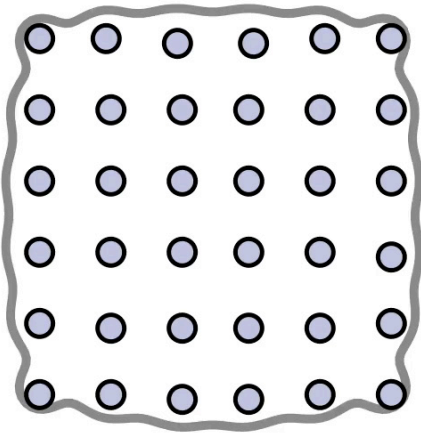
shaded



target

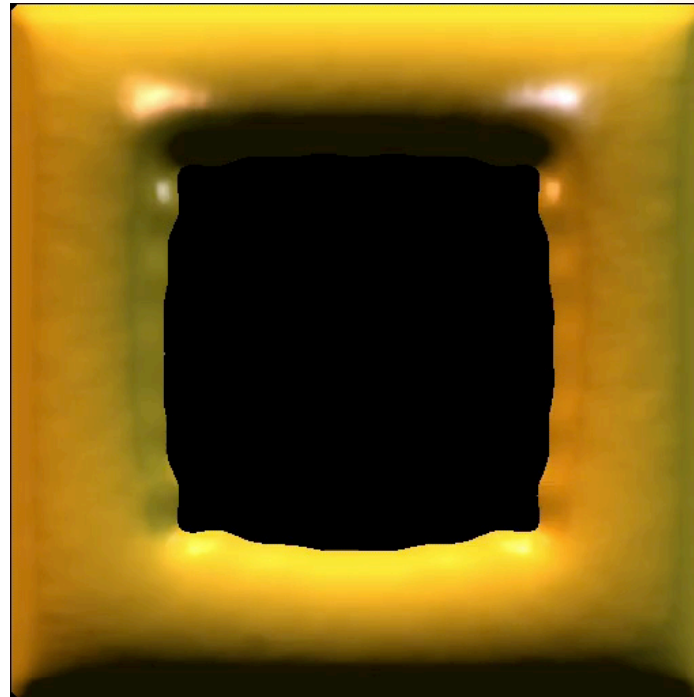
# Inflatable surfaces

implicit surface



geometry

optimized



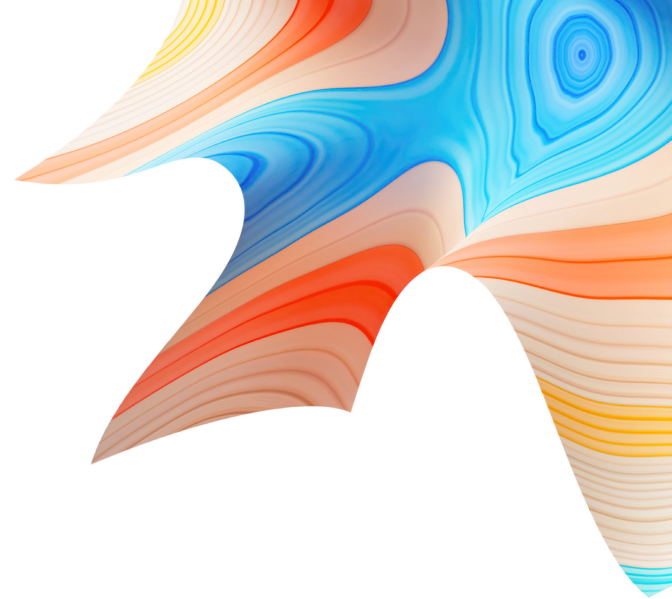
shaded

target





# what's next?



# Generalizing to new boundary conditions

primal PDE

$$\Delta u = 0 \quad \text{on } \Omega(\pi)$$

$$\frac{\partial u}{\partial n} = h \quad \text{on } \partial\Omega(\pi)$$

differential PDE

$$\Delta \dot{u} = 0 \quad \text{on } \Omega(\pi)$$

$$\frac{\partial \dot{u}}{\partial n} = V_n \left( \frac{\partial h}{\partial n} - \frac{\partial^2 u}{\partial n^2} \right) + \langle \nabla u, \nabla_{\Gamma} V_n \rangle \quad \text{on } \partial\Omega(\pi)$$

Requires higher order boundary gradients estimate + more branching

recursive control variates

sample reuse with RESTIR

gradient filtering

## Recursive Control Variates for Inverse Rendering

BAPTISTE NICOLET, *École Polytechnique Fédérale de Lausanne (EPFL) and NVIDIA, Switzerland*  
 FABRICE ROUSSELLE, *NVIDIA, Switzerland*  
 JAN NOVÁK, *NVIDIA, Czech Republic*  
 ALEXANDER KELLER, *NVIDIA, Germany*  
 WENZEL JAKOB, *École Polytechnique Fédérale de Lausanne (EPFL), Switzerland*  
 THOMAS MÜLLER, *NVIDIA, Switzerland*



Fig. 1. We improve physics-based differentiable rendering by introducing a recursive control variate into the optimization loop. Left: A baseline method [Vicini et al. 2023] repeatedly renders noisy images to compute gradients. The noise leads to poor convergence. Right: applied on top of the baseline, our recursive control variate reduces the noise using information from similar renderings in prior optimization steps. This leads to a reconstruction that is much closer to the reference image. The plots show the evolution of the loss function and rendering variance as the optimization progresses.

We present a method for reducing errors—variance and bias—in physically-based differentiable rendering (PDR). Typical applications of PDR repeatedly render a scene as part of an optimization loop involving gradient descent. The actual change introduced by each gradient descent step is often relatively small, causing a significant degree of redundancy in this computation. We exploit this redundancy by formulating a gradient estimator that employs a recursive control variate, which leverages information from previous optimization steps. The control variate reduces variance in gradients, and, perhaps more importantly, alleviates issues that arise from differentiating loss functions with respect to noisy inputs, a common cause of drift to bad local minima or divergent optimizations. We experimentally evaluate our approach on a variety of path-traced scenes containing surfaces and volumes and observe that primal rendering efficiency improves by a factor of up to 10.

CCS Concepts • Computing methodologies → Rendering

Authors' address: Baptiste Nicolet, Ecole Polytechnique Fédérale de Lausanne (EPFL) and NVIDIA, Switzerland, baptiste.nicolet@epfl.ch; Fabrice Rousselet, NVIDIA, Switzerland, frousselet@nvidia.com; Jan Novák, NVIDIA, Czech Republic, jan.novak@nvidia.com; Alexander Keller, NVIDIA, Germany, akeller@nvidia.com; Wenzel Jakob, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, wenzel.jakob@epfl.ch; Thomas Müller, NVIDIA, Switzerland, thomas@nvidia.com.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. Copying for other than research, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Requested permissions from permissions@acm.org.

© 2023 Copyright held by the owner(s). Publication rights licensed to ACM. 0730-0297/2023/12-ART214-12. \$15.00.  
<https://doi.org/10.1145/3592139>

ACM Trans. Graph., Vol. 42, No. 4, Article 1. Publication date: August 2023.

[Nicolet et al. 2023]

## Amortizing Samples in Physics-Based Inverse Rendering using ReSTIR

YU-CHEN WANG, *University of California, Irvine, USA*  
 CHRIS WYMAN, *NVIDIA, USA*  
 LI-FAN WU, *NVIDIA, USA*  
 SHUANG ZHAO, *University of California, Irvine, USA*

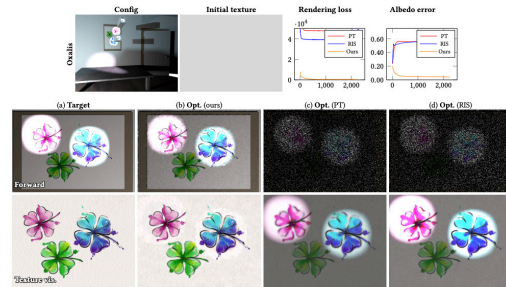


Fig. 1. Our new technique reuses temporal data in the context of physics-based inverse direct illumination. During inverse rendering, when optimizing a scene iteratively using gradient-based methods such as stochastic gradient descent or Adam [Kingma and Ba 2015], we reuse light samples spatially and temporally (across iterations), offering significantly cleaner forward rendering and gradient estimates than baseline methods without reuse. This example uses the “Owlie” painting lit by several bright spot lights and a dim fill light. We optimize the spatially varying albedo (initialized using the gray texture shown) of the painting. (c, d) The baseline methods PT (B) and ReSTIR (B) produce high variance—especially in dark areas only lit by the fill light, causing highly biased results. (b) Our method, on the other hand, enjoys significantly more accurate reconstructions.

Recently, great progress has been made in physics-based differentiable rendering. Existing differentiable rendering techniques typically focus on static scenes, but during inverse rendering—a key application for differentiable rendering—the scene is updated dynamically by each gradient step. In this paper, we propose a novel technique to amortize samples across iterations.

Authors' address: Yu Chen Wang, yucheng@uci.edu, University of California, Irvine, USA; Chris Wyman, cwyman@nvidia.com, NVIDIA, USA; Li-Fan Wu, li-fan@nvidia.com, NVIDIA, USA; Shuang Zhao, shuangz@uci.edu, University of California, Irvine, USA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for third party components of this work must be honored. For all other uses, contact the owner(s).

© 2023 Copyright held by the owner(s). Publication rights licensed to ACM. 0730-0297/2023/12-ART214-12. \$15.00.  
<https://doi.org/10.1145/3518331>

ACM Trans. Graph., Vol. 42, No. 4, Article 214. Publication date: December 2023.

[Wang et al. 2023]

## Spatiotemporal Bilateral Gradient Filtering for Inverse Rendering

WESLEY CHANG<sup>1</sup>, *University of California San Diego, USA*  
 XUANDA YANG<sup>1</sup>, *University of California San Diego, USA*  
 YASH BELHE<sup>1</sup>, *University of California San Diego, USA*  
 RAVI RAMAMOORTHY<sup>2</sup>, *University of California San Diego, USA*  
 TZU-MAO LI<sup>1</sup>, *University of California San Diego, USA*

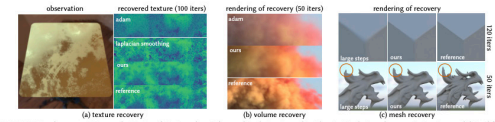


Fig. 1. We introduce a spatiotemporal optimizer that generalizes Adam and Laplacian Smoothing (Large Steps). It applies an anisotropic cross-bilateral filter to the gradient across space. In addition to temporal filtering (like Adam), our cross-bilateral filter reduces gradient noise and improves conditioning for anisotropic objectives by imposing a piecewise smoothness prior. Our method enables faster convergence and higher quality inverse rendering of (a) textures, (b) volumes and (c) meshes at very low sample counts, all experiments only use 1 sample per pixel for gradient estimation. (a) For roughness texture recovery after 100 iterations, our method has converged while others have artifacts. (b) For volume density and albedo recovery in just 50 iterations, our method can already recover the rough shape and color; further optimization with higher sample counts recovers details. (c) For mesh recovery our method is able to recover sharp features (top row, cube) and thin structures (bottom row, dragon) faster than competing methods. Scenes adapted from At the Window ©Bernd Vogel, Autumn Field ©Janis Gunt and Sergej Mjomboda, High-Rise Smoke Plane ©Janis Gunt, Kloppestein 06 ©Gert Zank and Anan Dragon ©Stanford Computer Graphics Laboratory.

In inverse rendering, gradient-based methods, which have seen great progress in the recent years, are typically used in conjunction with the Adam optimizer. While Adam usually improves convergence by temporally filtering gradients over previous iterations to reduce noise, it is not tailored to inverse rendering where the target signals (textures, volumes, or geometry) are usually piecewise smooth. Previous work has applied the inverse Laplacian operator to smooth gradients spatially, but this isotropic filtering can often lead to over-smoothing. We propose a spatiotemporal optimizer that can significantly speed up the convergence over Adam, by enforcing the optimization parameters updates to be piecewise smooth through a lightweight spatial-domain cross-bilateral filter. We discuss different options of combining spatial filtering and Adam’s temporal filtering, and provide intuitions for different scenarios. We show that our filtering leads to significantly higher-quality reconstructions in different inverse problems including texture, volume and geometry recovery.

CCS Concepts • Computing methodologies → Rendering

“The three authors contributed equally to this research.”

Authors' Contact Information: Wesley Chang, we22@ucsd.edu, University of California San Diego, USA; Xuanda Yang, xy2@ucsd.edu, University of California San Diego, USA; Yash Belhe, ybelhe@ucsd.edu, University of California San Diego, USA; Ravi Ramamoorthy, rramamoorthy@ucsd.edu, University of California San Diego, USA; Tzu-Mao Li, liz@ucsd.edu, University of California San Diego, USA.

SA Conference Papers '24, December 3–4, 2024, Tokyo, Japan  
 © 2024 Copyright held by the owner(s).  
 This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in SIGGRAPH Asia 2024 Conference Papers (SA Conference Papers '24), December 3–4, 2024, Tokyo, Japan. <https://doi.org/10.1145/3605268.3607066>.

Additional Key Words and Phrases: differentiable rendering, inverse rendering, optimization, preconditioning, bilateral filtering.

ACM Reference Format: Wesley Chang, Xuanda Yang, Yash Belhe, Ravi Ramamoorthy, and Tzu-Mao Li. 2024. Spatiotemporal Bilateral Gradient Filtering for Inverse Rendering. In SIGGRAPH Asia 2024 Conference Papers (SA Conference Papers '24), December 3–4, 2024, Tokyo, Japan. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3605268.3607066>

## 1 INTRODUCTION

Gradient-based optimization has enabled many inverse rendering applications such as texture, volume, and geometry recovery from observed images. With a few exceptions, most inverse rendering works use the Adam optimizer [Kingma and Ba 2015] to process gradients and update optimization parameters. In this work, we show that we can significantly speed up Adam and other prior work [Nicolet et al. 2021; Osler et al. 2018] over a wide range of inverse rendering tasks, by applying edge-aware spatial filtering, such as a lightweight cross-bilateral filter [Haiman and Durand 2004; Petsochagg et al. 2004] at each iteration, as seen in Figure 1.

Adam can be seen as a temporal filter that rescales the gradient component-wise based on its value from previous iterations. This has two major benefits: first, when the gradient is noisy, due to either the stochastic evaluation of the objective function and its gradient or combing; temporal filtering reduces noise. Second, it adjusts the learning rate per component to use faster learning rates on gradient components that change slower over time, and vice versa.

SA Conference Papers '24, December 3–4, 2024, Tokyo, Japan.

project



code



**Thank you!**

project page: [imaging.cs.cmu.edu/differential\\_walk\\_on\\_spheres](https://imaging.cs.cmu.edu/differential_walk_on_spheres)

code: [github.com/baileymiller/differential\\_wos](https://github.com/baileymiller/differential_wos)



**SIGGRAPH 東京**  
**ASIA 2024**  
**TOKYO**