Monte Carlo Geometry Processing

Rohan Sawhney, *NVIDIA*
Bailey Miller, *Carnegie Mellon*

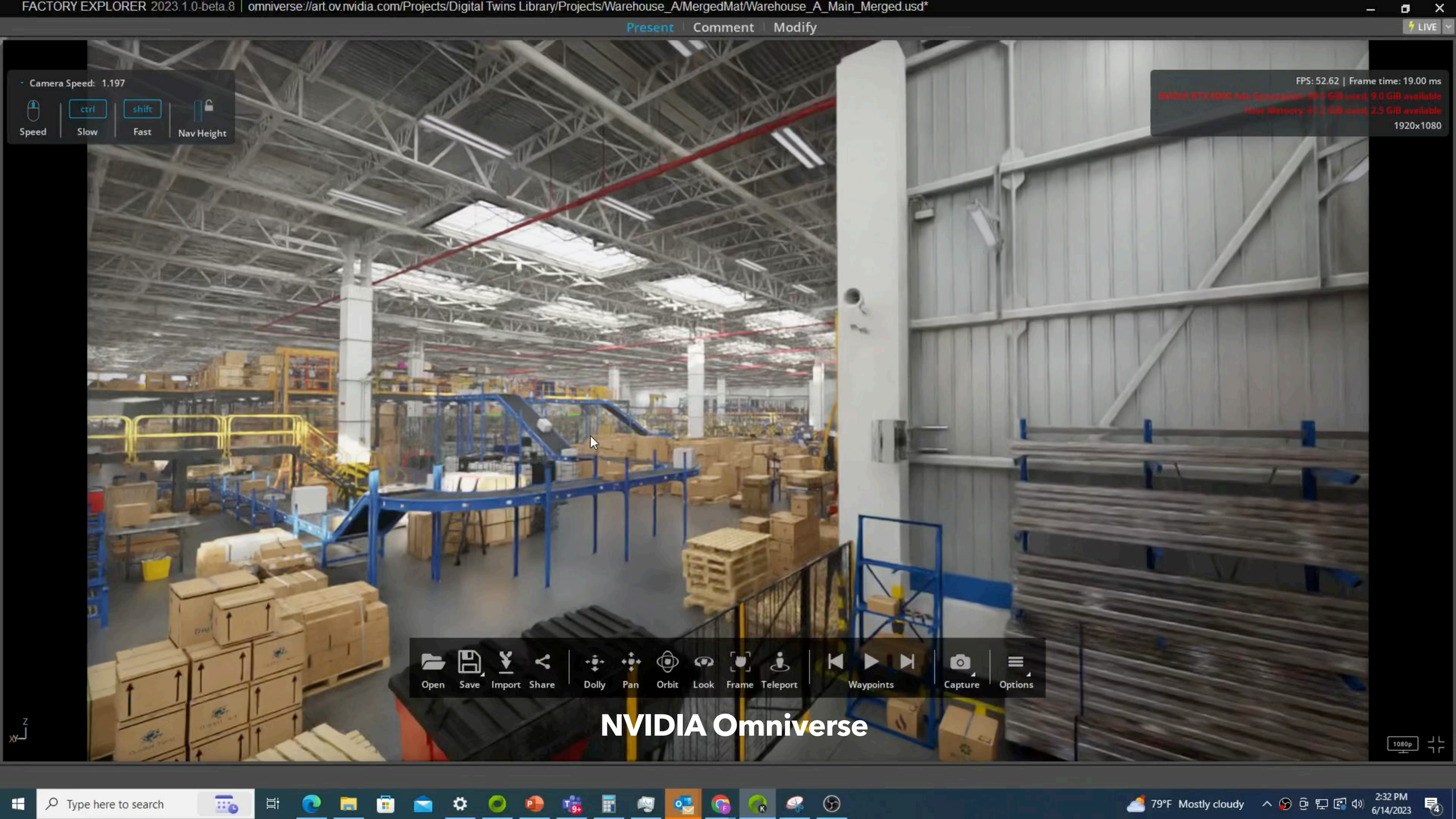
Presenters (and their advisors)

Rohan Sawhney
NVIDIA

Bailey Miller
CMU PhD

Ioannis Gkioulekas
CMU

Keenan Crane
CMU


FACTORY EXPLORER 2023.1.0-beta.8 | omniverse://art.ov.nvidia.com/Projects/Digital Twins Library/Projects/Warehouse_A/MergedMat/Warehouse_A_Main_Merged.usd*
Present
Comment
Modify
LIVE
Camera Speed: 1.197
Speed
ctrl
Slow
shift
Fast
Nav Height
FPS: 52.62 | Frame time: 19.00 ms
1920x1080
Open
Save
Import
Share
Dolly
Pan
Orbit
Look
Frame
Teleport
Waypoints
Capture
Options
NVIDIA Omniverse
1080p
Type here to search
79°F Mostly cloudy
2:32 PM
6/14/2023

Kit RTX Dev App 105.0+master.0.0946391b.local - omniverse://art-sc-ro.ov.nvidia.com/Projects/RacerX/Scenes/RacerX_Master/RacerX_Master_v2.usda*

File Edit Create Window Rendering Profiler Help

LIVE CACHE: OFF

Viewport

RTX - Real-Time | Perspective

Camera Speed: 0.676

Speed | ctrl Slow | shift Fast

FPS: 29.13, Frame time: 34.33 ms
NVIDIA GeForce RTX 4090: 15.6 GiB used, 5.0 GiB available
Host Memory: 57.1 GiB used, 70.8 GiB available
2560x1440

Console | Content

Import | omniverse://art-sc-ro.ov.nvidia.com/Projects/RacerX/Scenes/RacerX_Master/

- RacerX_Audio
- RacerX_Candles
- RacerX_Cinematic
- RacerX_DinoPark
- RacerX_Driveway
- RacerX_Dust_Env
- RacerX_Garage
- RacerX_Master
- RacerX_Shots
- RacerX_WestsideBackStre
- RaceX_Dust_Pit
- Temp
- TestLevels

Name	Date	Size
_the_scene_02	05/24/2022 08:57AM	
DO NOT USE	06/24/2022 01:04PM	
FX	06/20/2022 05:36AM	
LayeredMaterial	06/17/2022 03:13AM	
Lighting	06/16/2022 10:52AM	
PathFollowing	07/14/2022 12:51AM	
RCCar	06/17/2022 03:12AM	
Sublayers	07/31/2022 01:18PM	
TestLevels	06/20/2022 02:20AM	
audio_test.usda	07/27/2022 12:34PM	3.93 KB
cameras.usd	08/01/2022 03:06PM	1.89 KB
RacerX_Audio.usd	07/22/2022 07:51AM	

RacerX_Master
Date Modified 05/24/2022 08:56AM
Created by nusov@nvidia.com
Modified by nusov@nvidia.com
File size 0.00 KB

Checkpoints

Stage | Layer | Render Settings

Search

Name (Old to New)	Type
RT_Book_M13_01	Xform
RT_ShoeBox_A3_02	Xform
RT_ShoeBox_A3_01	Xform
RT_Book_A2_01	Xform
RT_ShoeBox_A2_01	Xform
RT_Book_C2_1	Xform
RT_WoodenBuildingBlock_A5_01	Xform
SM_CardboardSheets_A10_1	Xform
SM_CardboardSheets_A29_11	Xform
RT_DuctTapeRoll_A1_01	Xform
Jenga_RT	Scope
RT_CardboardSheets_A10_06	Xform
RT_CardboardSheets_A14_03	Xform
RT_CardboardSheets_A14_1	Xform
Lighting	Scope
Overcast	Scope
Sunset	Scope
Gameplay	Scope
Desk_lamp_A	RectLight
Desk_lamp_B	RectLight
Desk_lamp_C	RectLight
Sunlight	DistantLight
Sky_portal_A	RectLight
Sky_portal_B	RectLight
Sky_portal_D	RectLight
Sky_portal_E	RectLight
Night	Scope
Light_blockers_ALWAYS_ON	Scope
Render	

Property

Add | Sunlight
Prim Path | /Lighting/Gameplay/Sunlight
Instanceable

Transform

	X	Y	Z
Translate	0.0	0.0	0.0
Rotate	26.87143	74.14741	106.00648
Scale	1.0	1.0	1.0

Light

Main

	R	G	B
Color	1.0	0.6	0.6

Enable Color Temperature
Color Temperature 5500.0
Intensity 1.0
Exposure 13.5
Normalize Power
Angle 1.0
Diffuse Multiplier 1.0
Specular Multiplier 1.0
Visible In Primary Ray
Disable Fog Interaction

Shaping

Light Link

EN 17°C Sunny ENG 4:28 PM 1/27/2023

DLSS 3 with Optical Multi Frame Generation

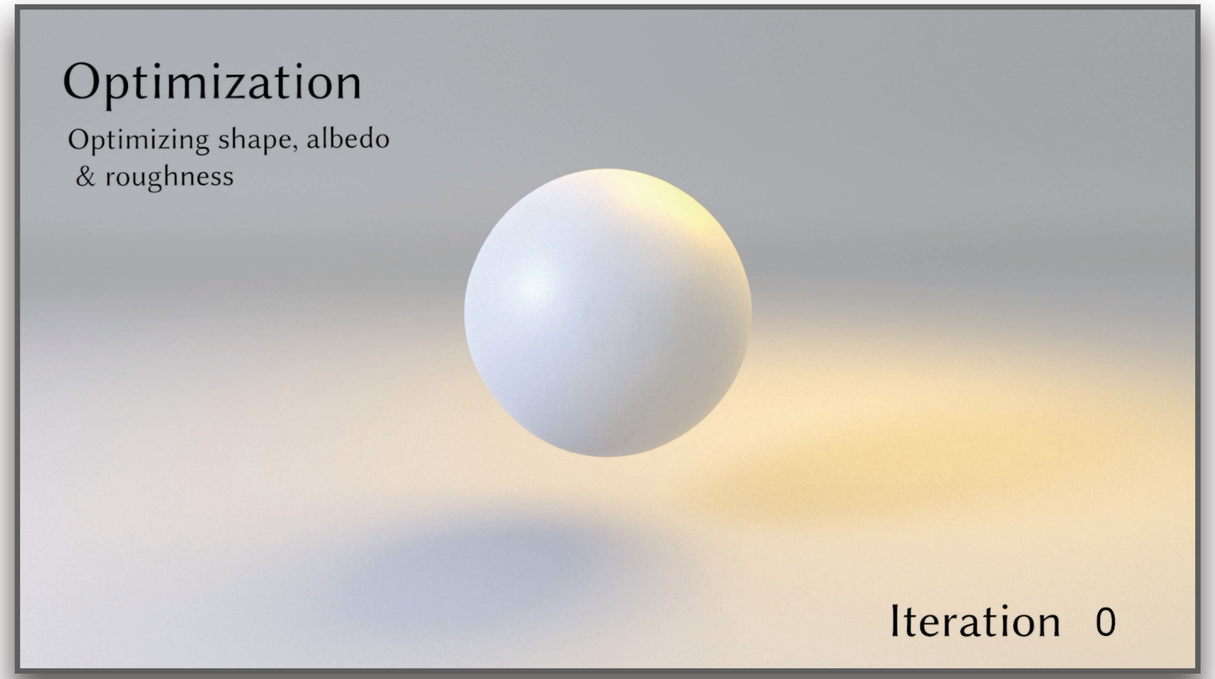
Boosts performance by using AI to generate more frames. DLSS analyzes sequential frames and motion data from the new Optical Flow Accelerator in GeForce RTX 40 Series GPUs to create additional high quality frames.

Photorealistic image generation has revolutionized industries

Entertainment

Pre-visualization

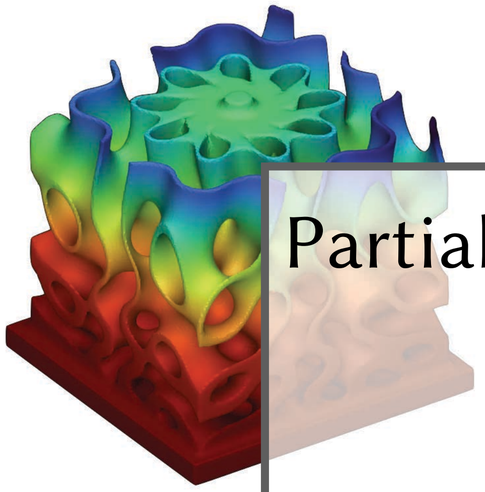
Photorealistic image generation will revolutionize more industries



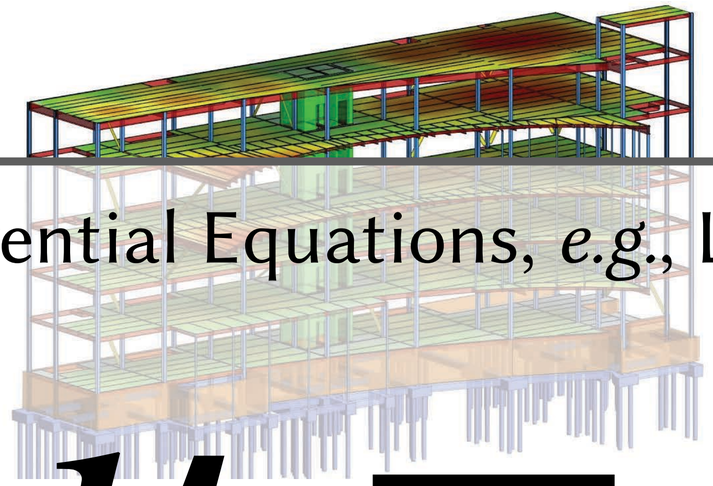
Computer Vision [Vicini et al, 2023]

Autonomous Driving [NVIDIA Drive Sim]

Physics beyond light transport



thermal diffusion



structural analysis

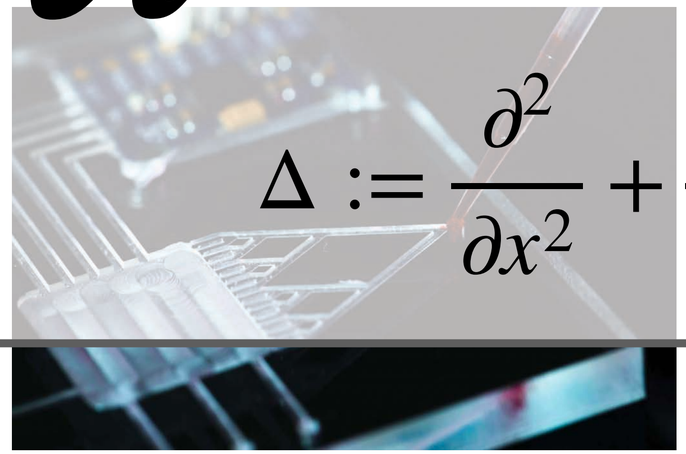
electrostatics

Partial Differential Equations, *e.g.*, Laplace Eq.

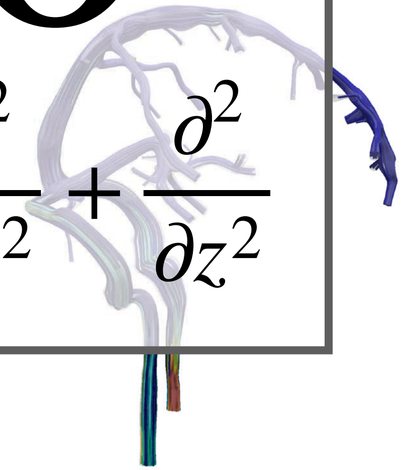
$$\Delta u = 0$$

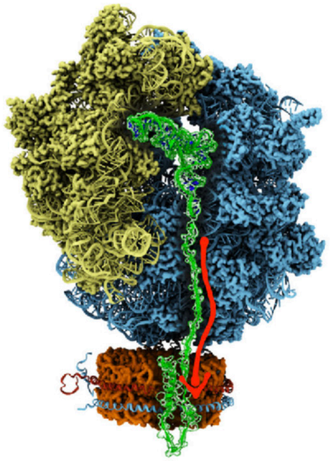
$$\Delta := \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

acoustic modeling



microfluidics



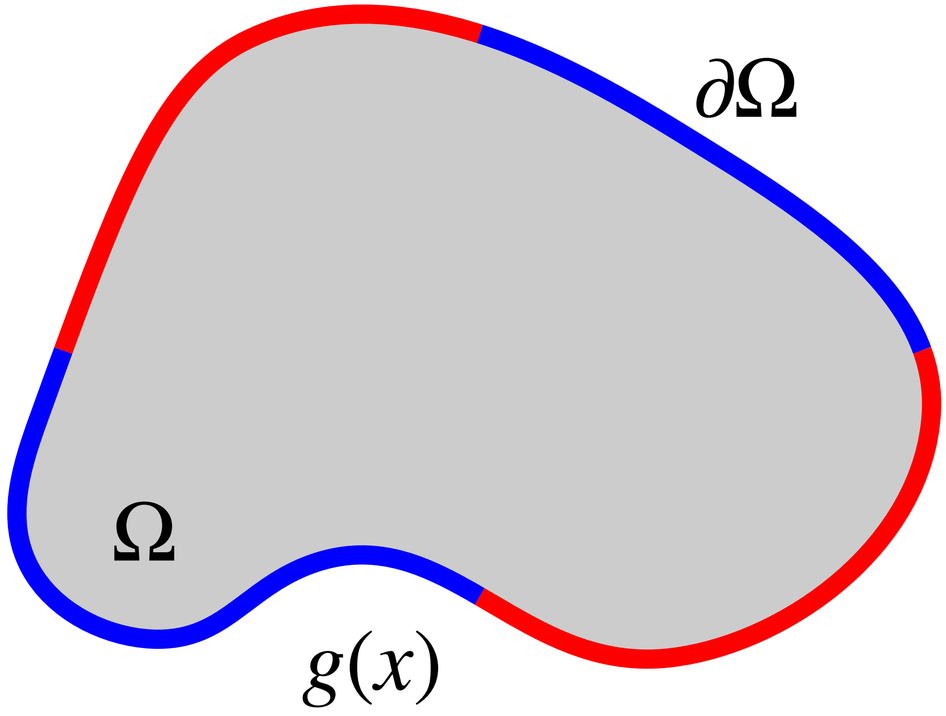


biophysics

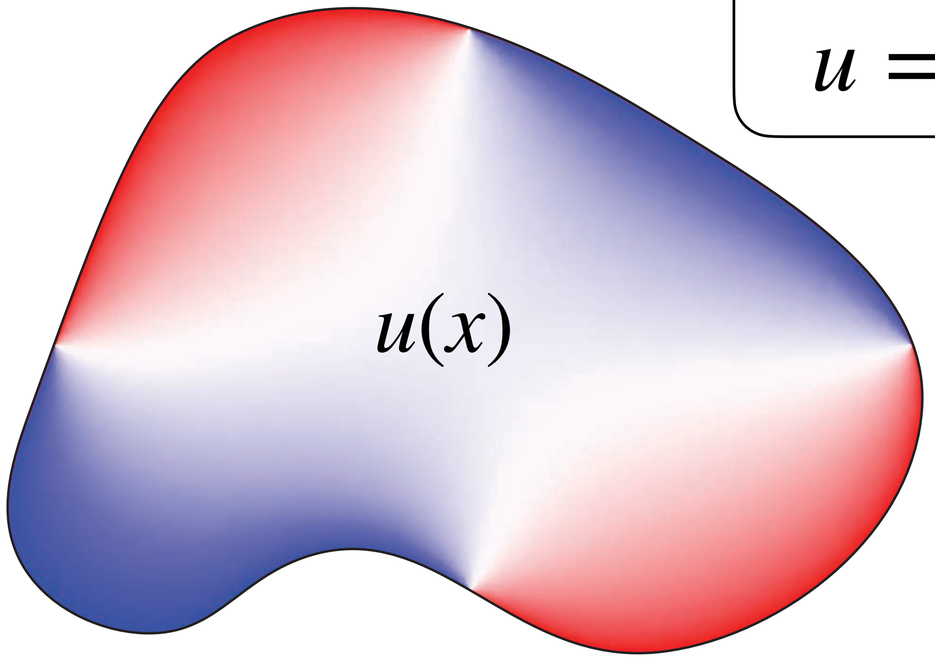
Partial Differential Equations (PDEs)

PDEs describe a function **implicitly** in terms of derivatives, solve to recover **explicit** function values

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

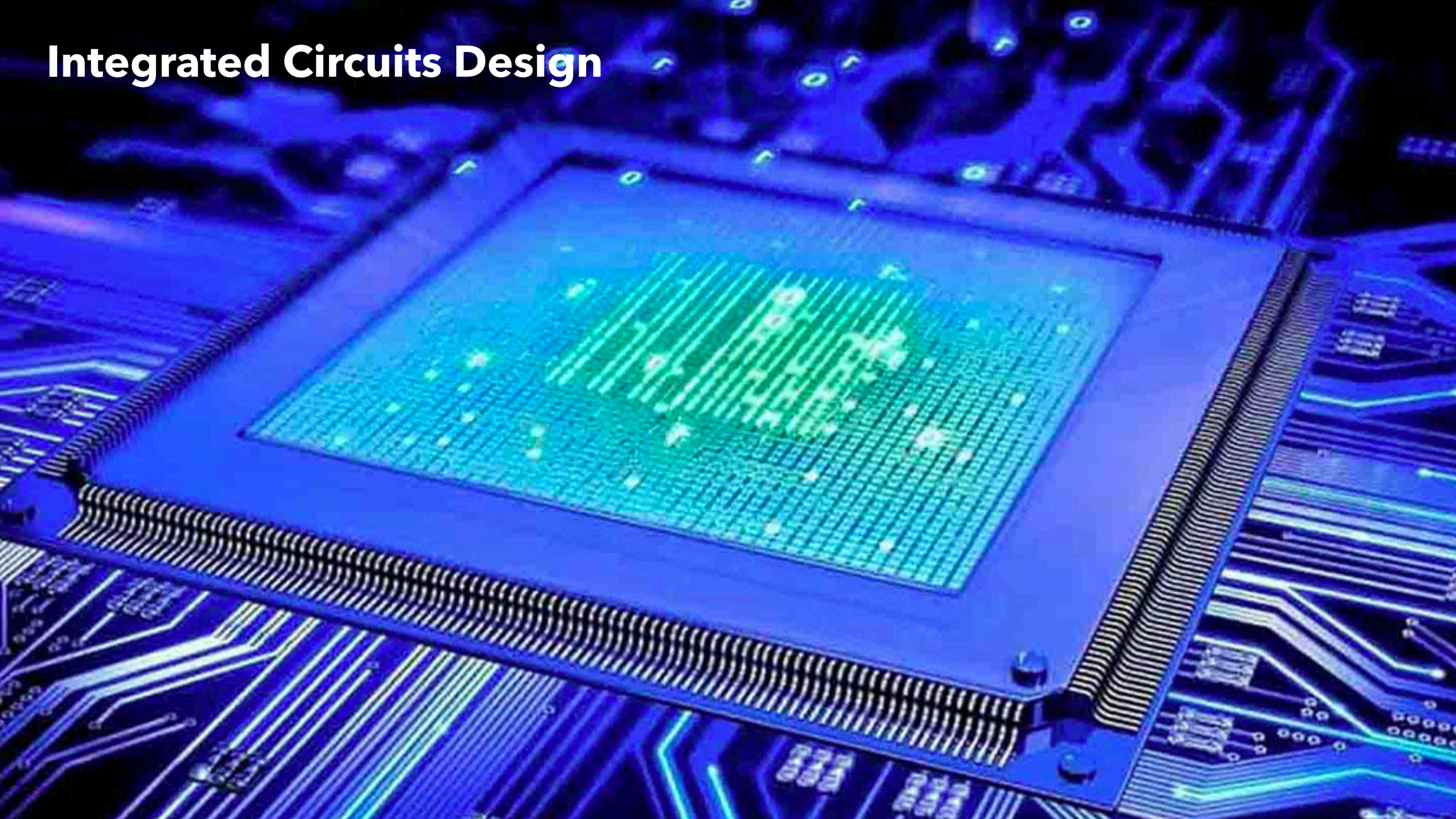


Given: values g on boundary

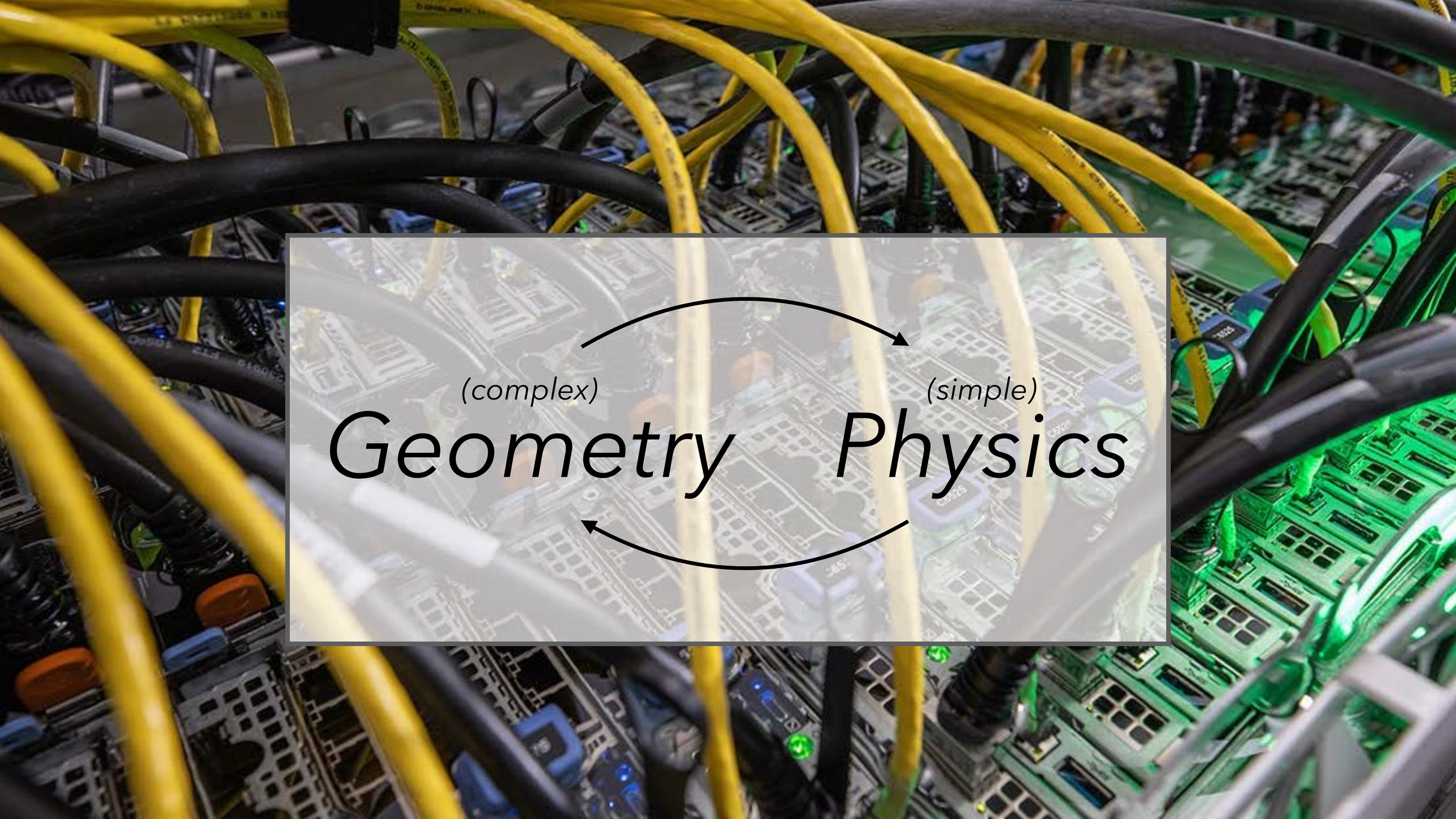


Find: solution u on interior

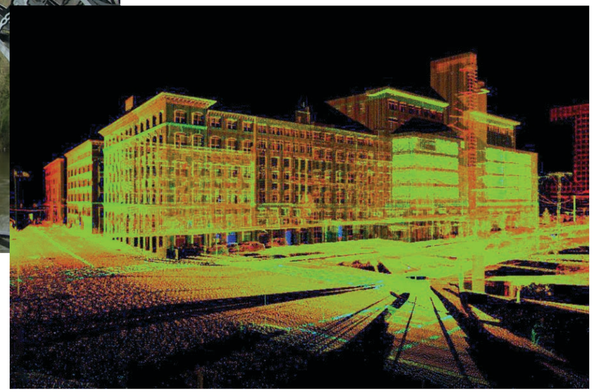
Integrated Circuits Design



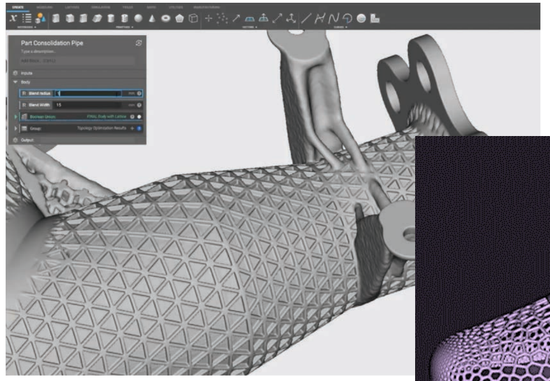
Data Center Design

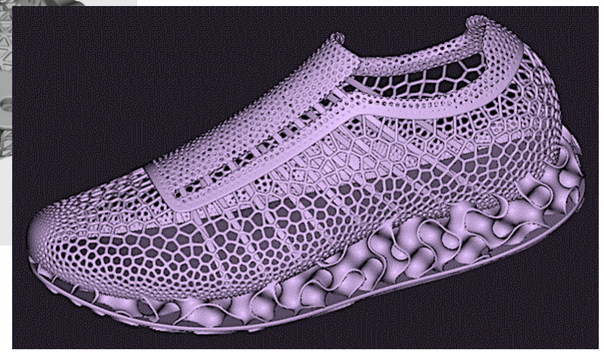

(complex)
Geometry
(simple)
Physics

Geometric complexity has increased drastically

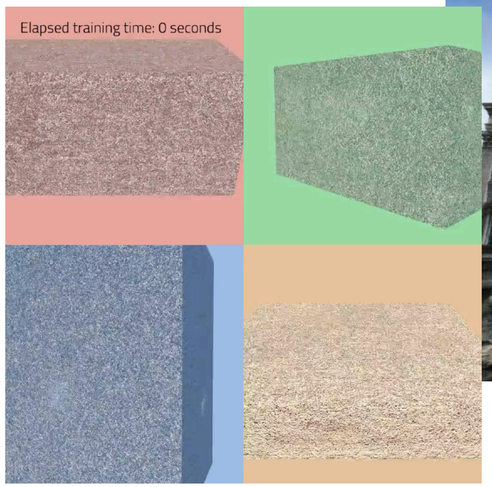


laser scanning / LiDAR

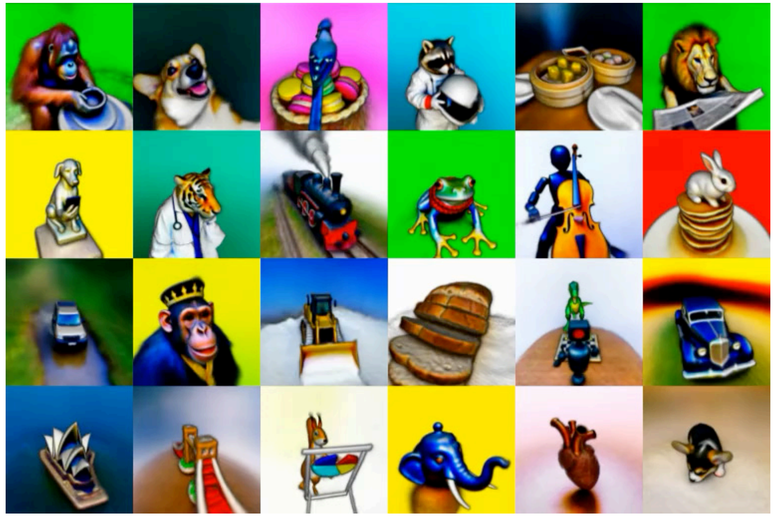




additive manufacturing



neural radiance fields



text-to-3D

Traditional methods for solving PDEs

"Zoo" of solvers:

– finite difference methods

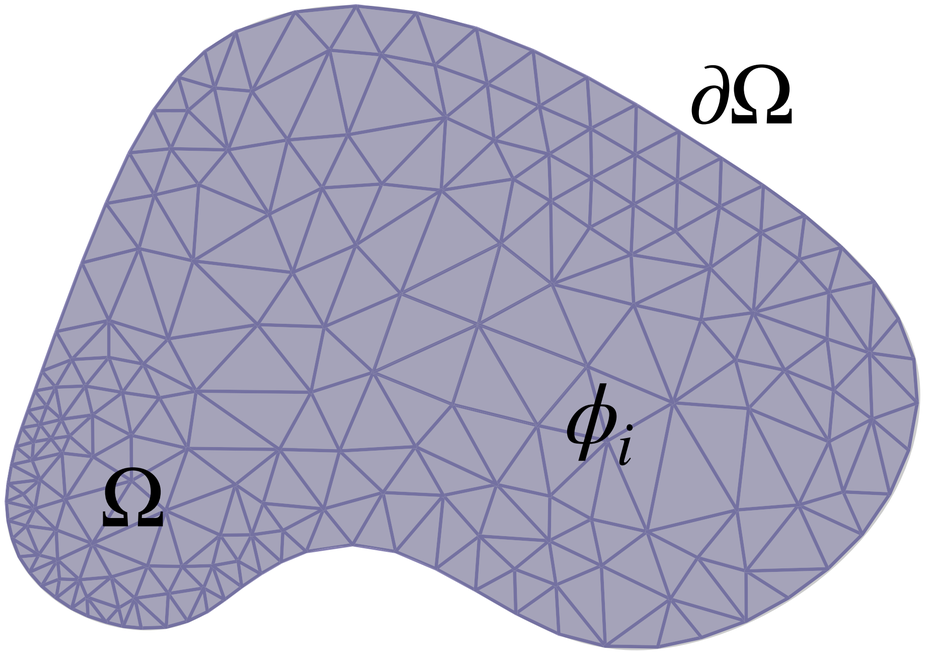
– finite element methods

– finite volume methods

– boundary element methods

– spectral methods

– …

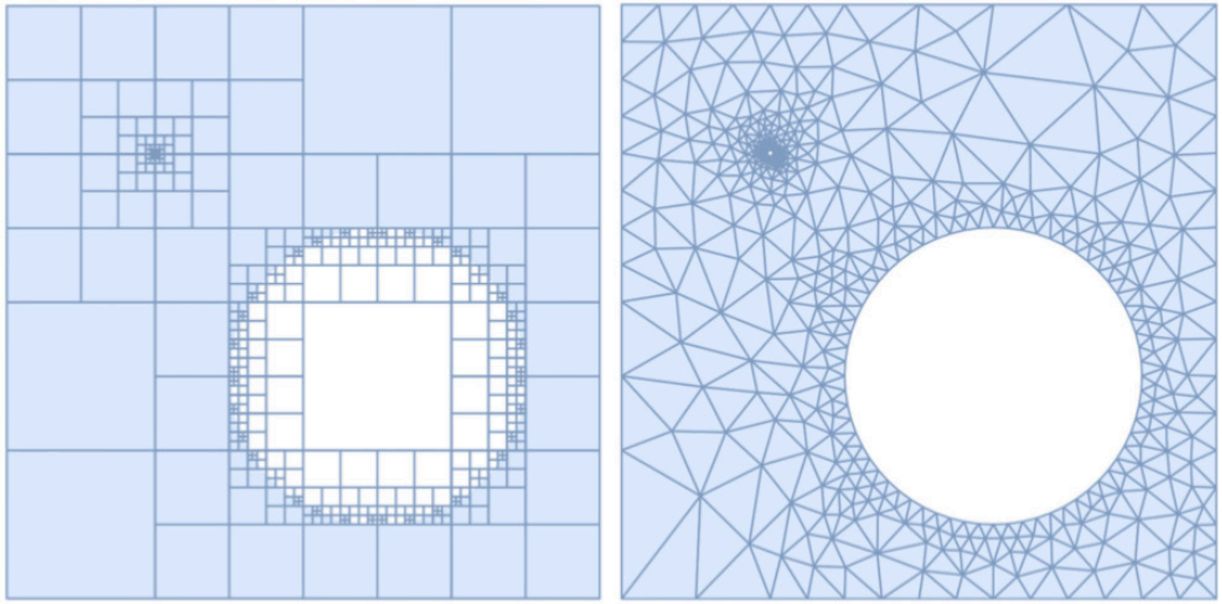


$$\boxed{\langle \Delta u, \phi_i \rangle = 0, \quad \forall i}$$

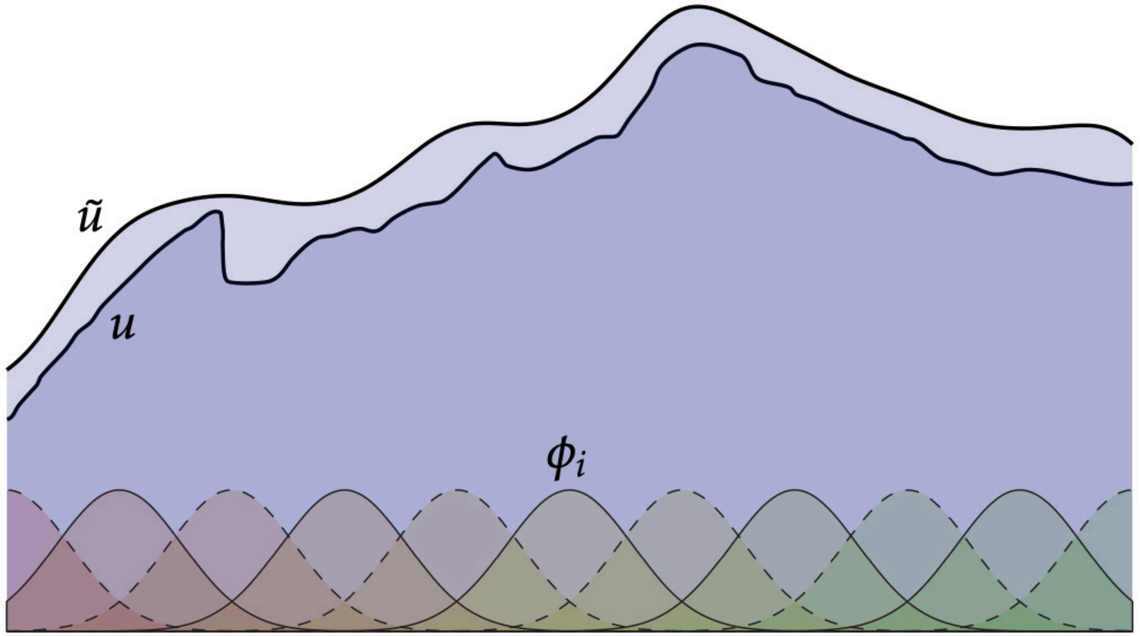
Common thread: all use *finite dimensional approximation*

Traditional methods for solving PDEs

Inevitable consequence of finite dimensional approximation:



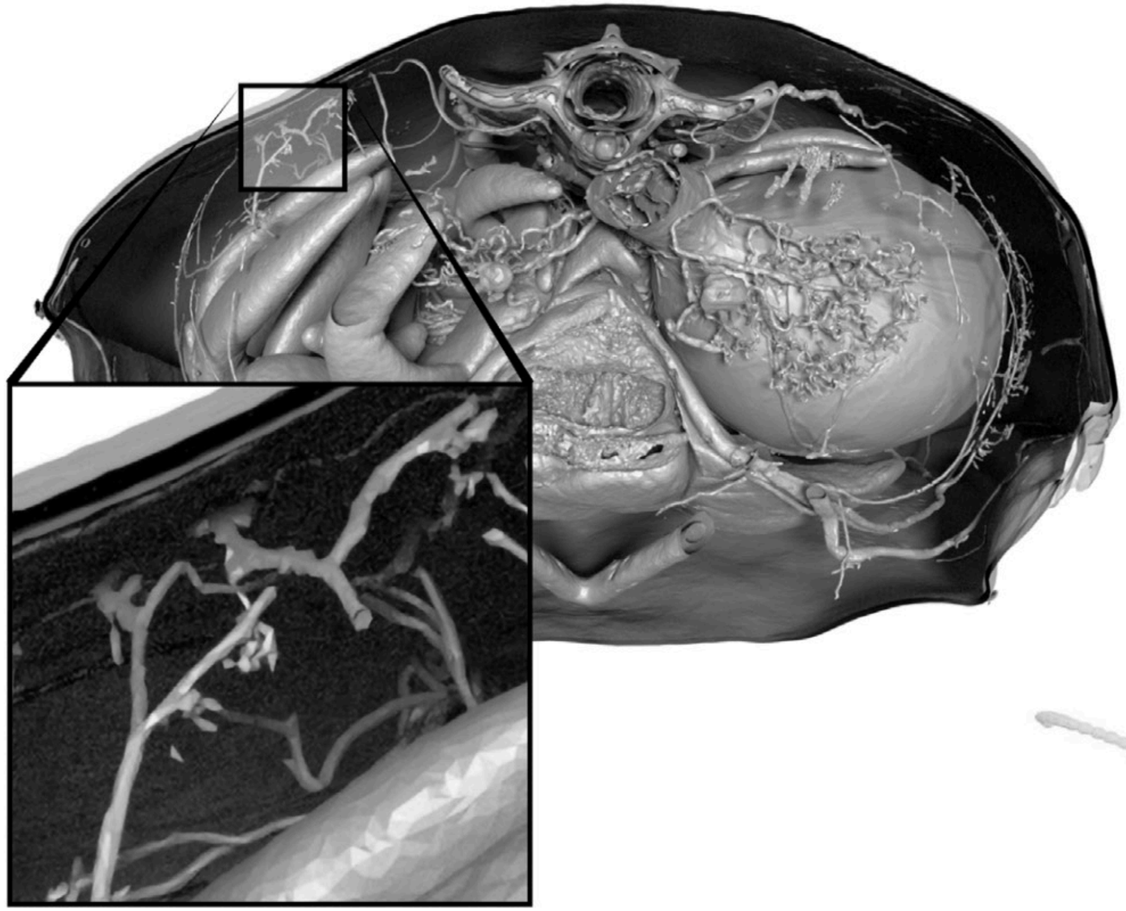
error due to approximation
of **geometry**

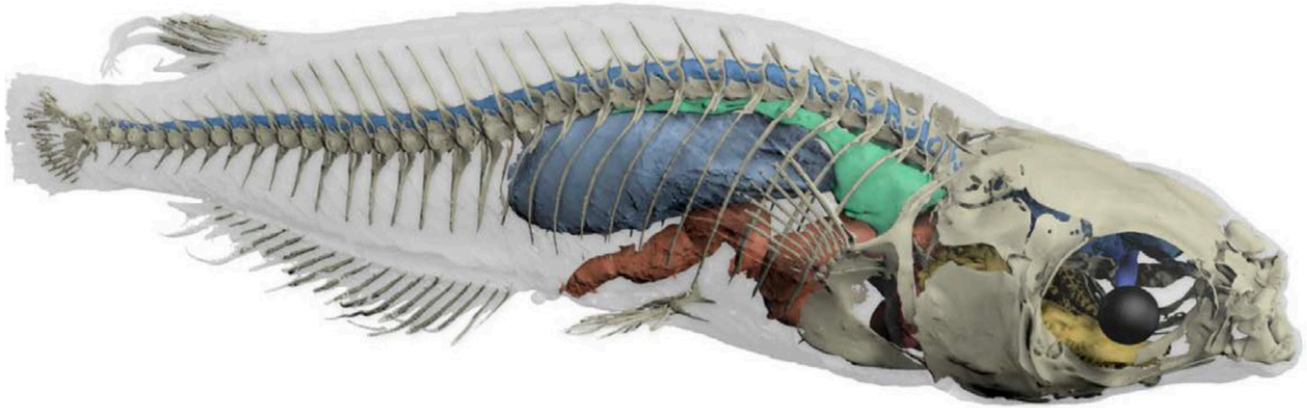


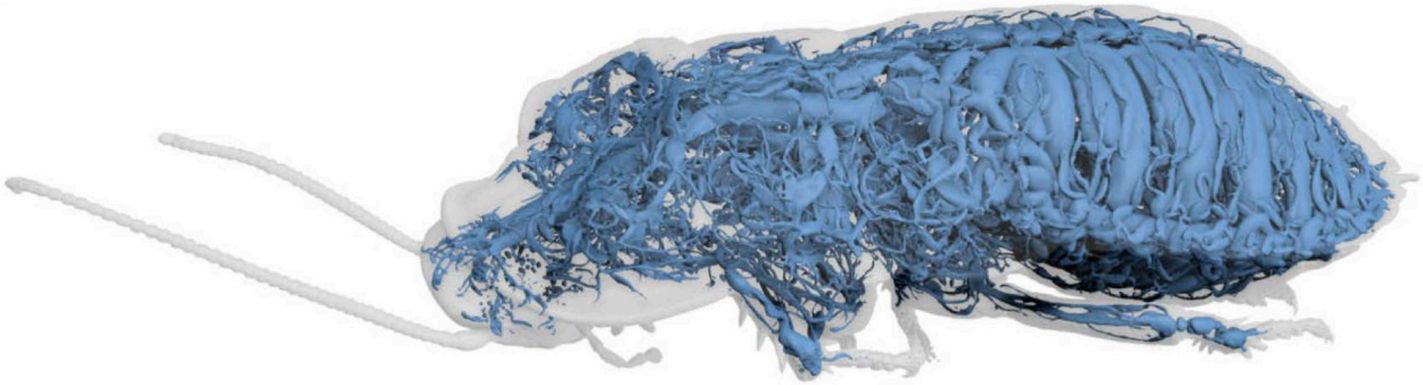
error due to approximation
of **functions**

Geometry found in physical world is extremely complex

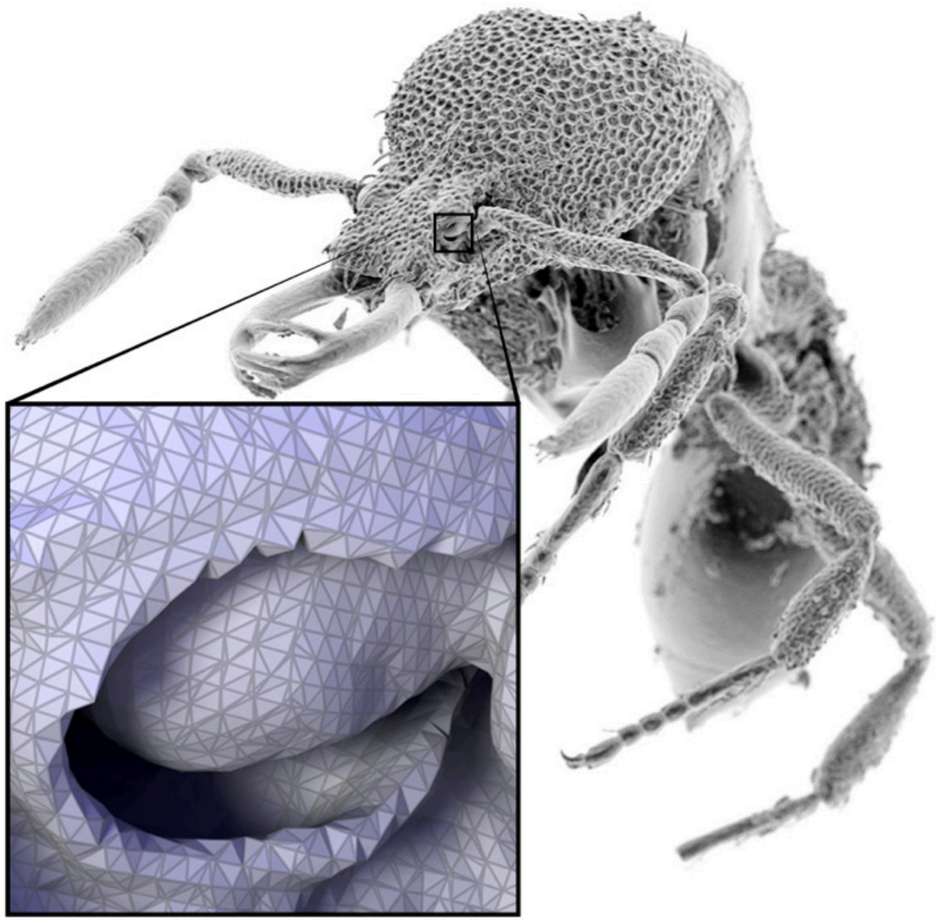
Example: high-resolution microCT scan







Complex geometry can take *extremely* long to mesh!

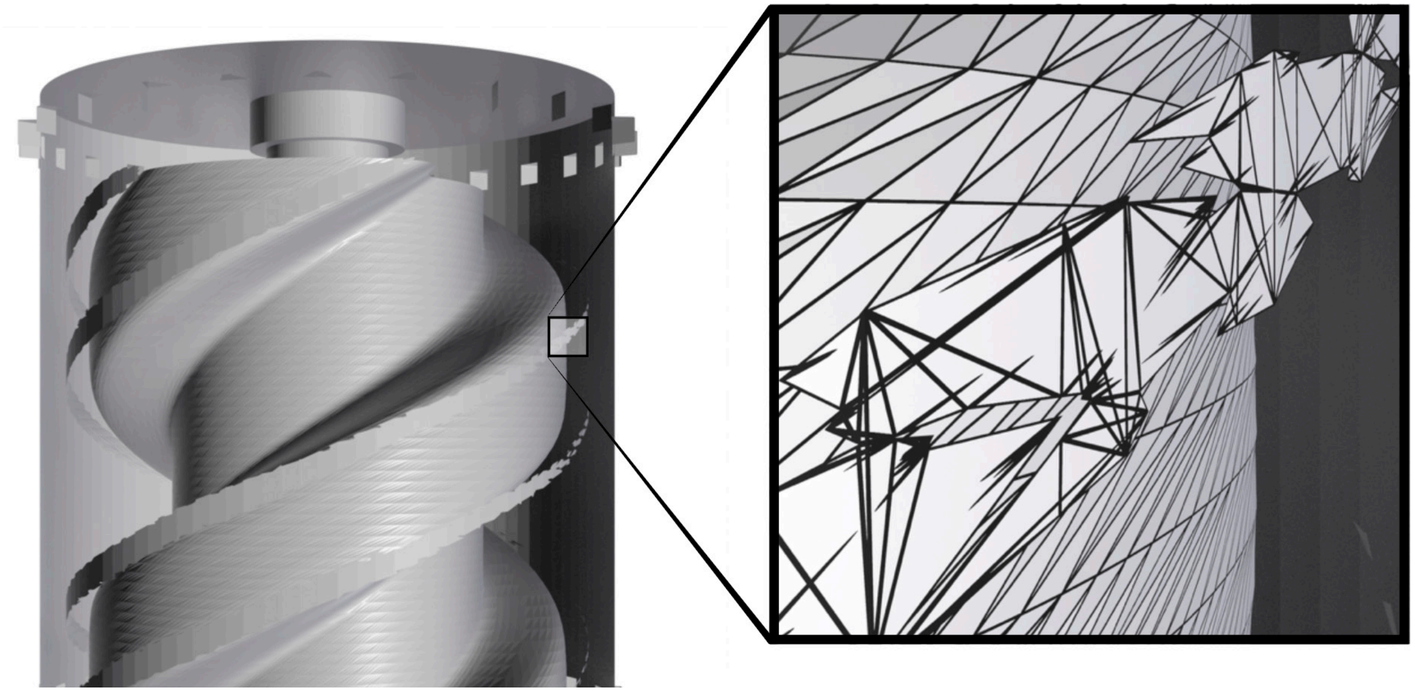


14 hours / 30 GB RAM

to generate "sim-ready" mesh

memory intensive & difficult to parallelize

Most geometry *in the wild* is not suitable for simulation



Error: could not mesh domain.

<u>Impossible</u> to run FEM solver.

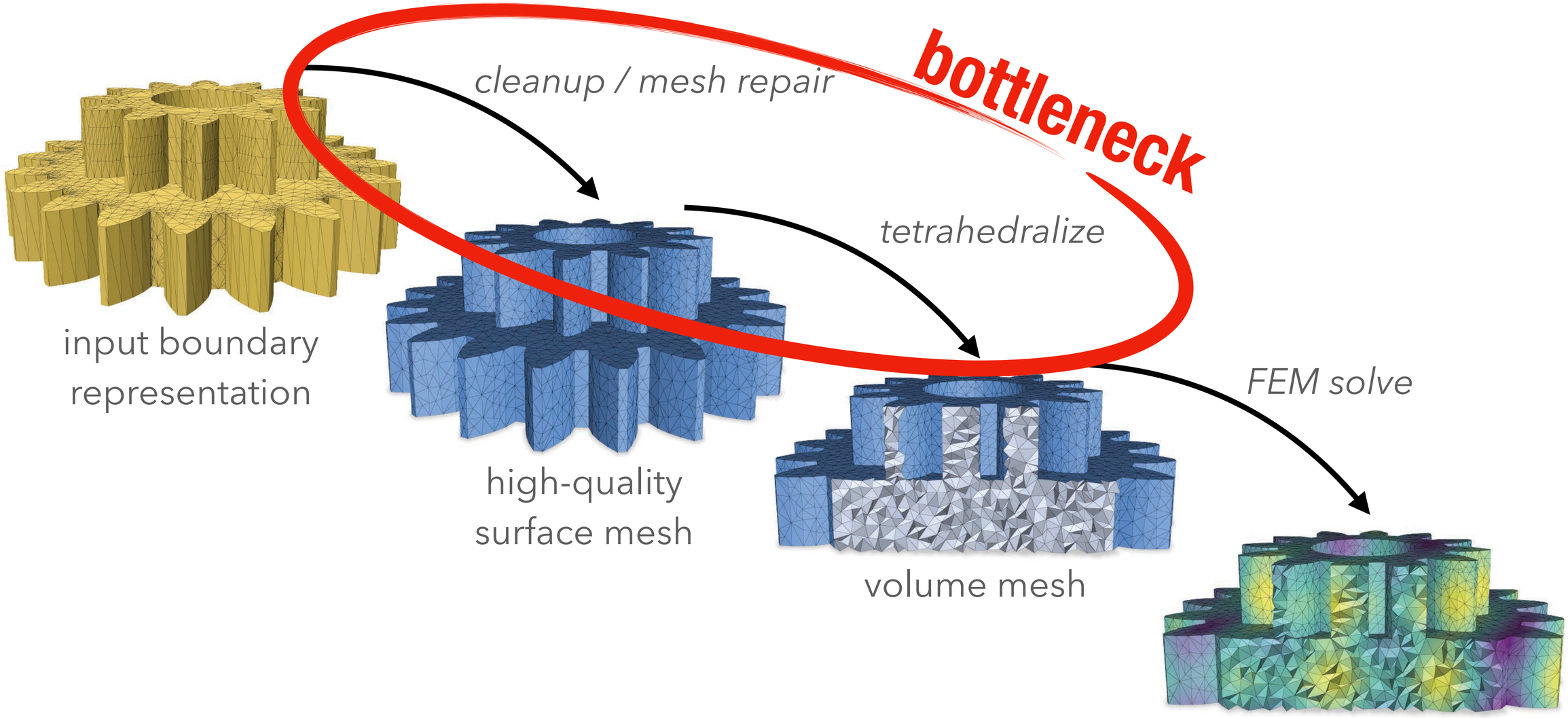
A bad mesh can yield a false impression of reality

Example: geodesic distance via FEM

FEM solution reference solution

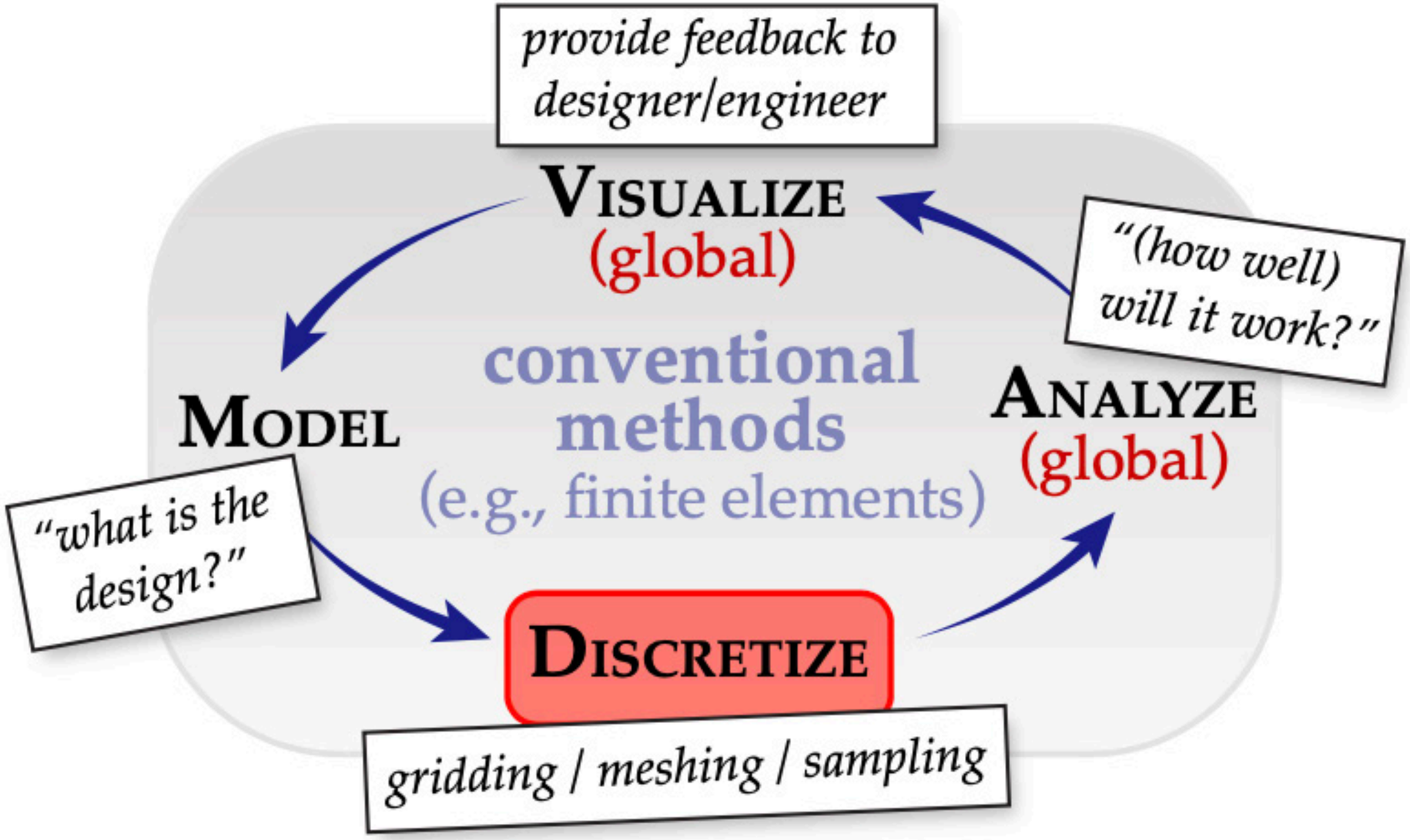
N. Sharp & K. Crane, A Laplacian for Nonmanifold Meshes (2020)

If meshing is slow, who cares if solver is fast?



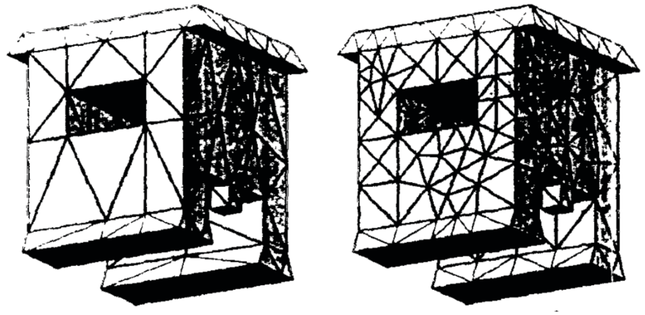
finite element method (FEM) pipeline

Meshing is *always* the bottleneck for simulation!

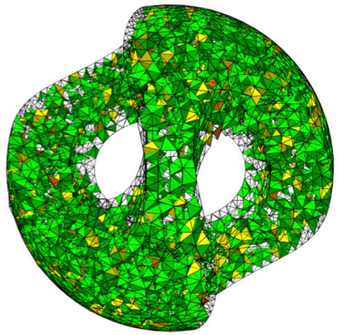


Robust meshing is still hard, even after 20+ years

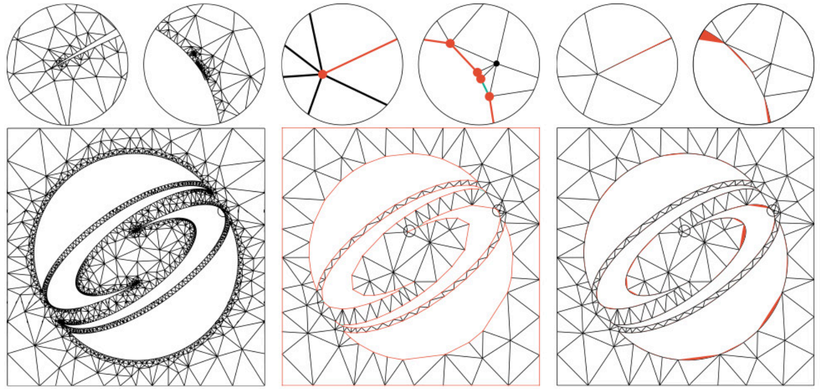
Not likely to **ever** be completely "solved":



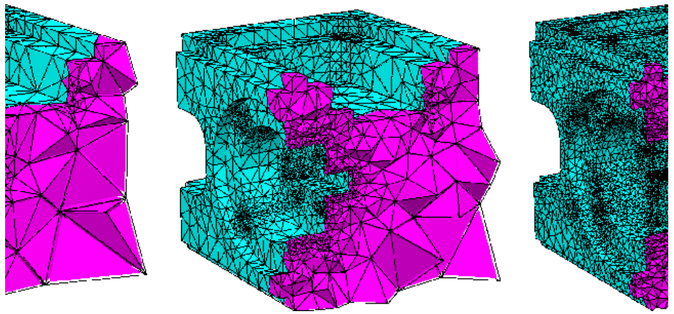
Tetrahedral Mesh Generation by Delaunay Refinement **[Shewchuk 1998]**



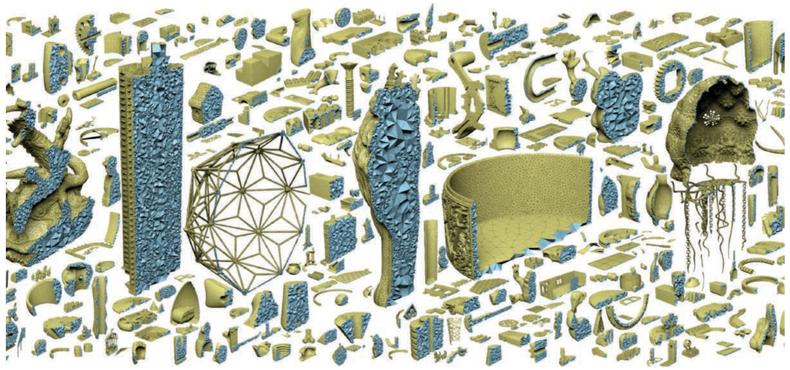
Robust Tetrahedral Meshing of Triangle Soups **[Spillman et al. 2006]**



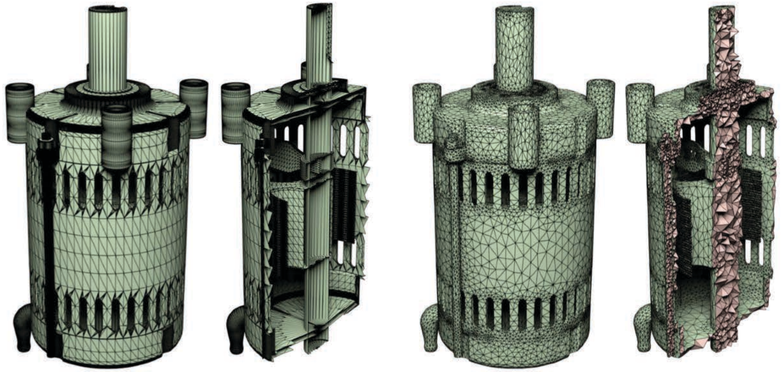
TriWild: Robust Triangulation With Curve Constraints **[Hu et al. 2019]**



A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator **[Si 2006]**

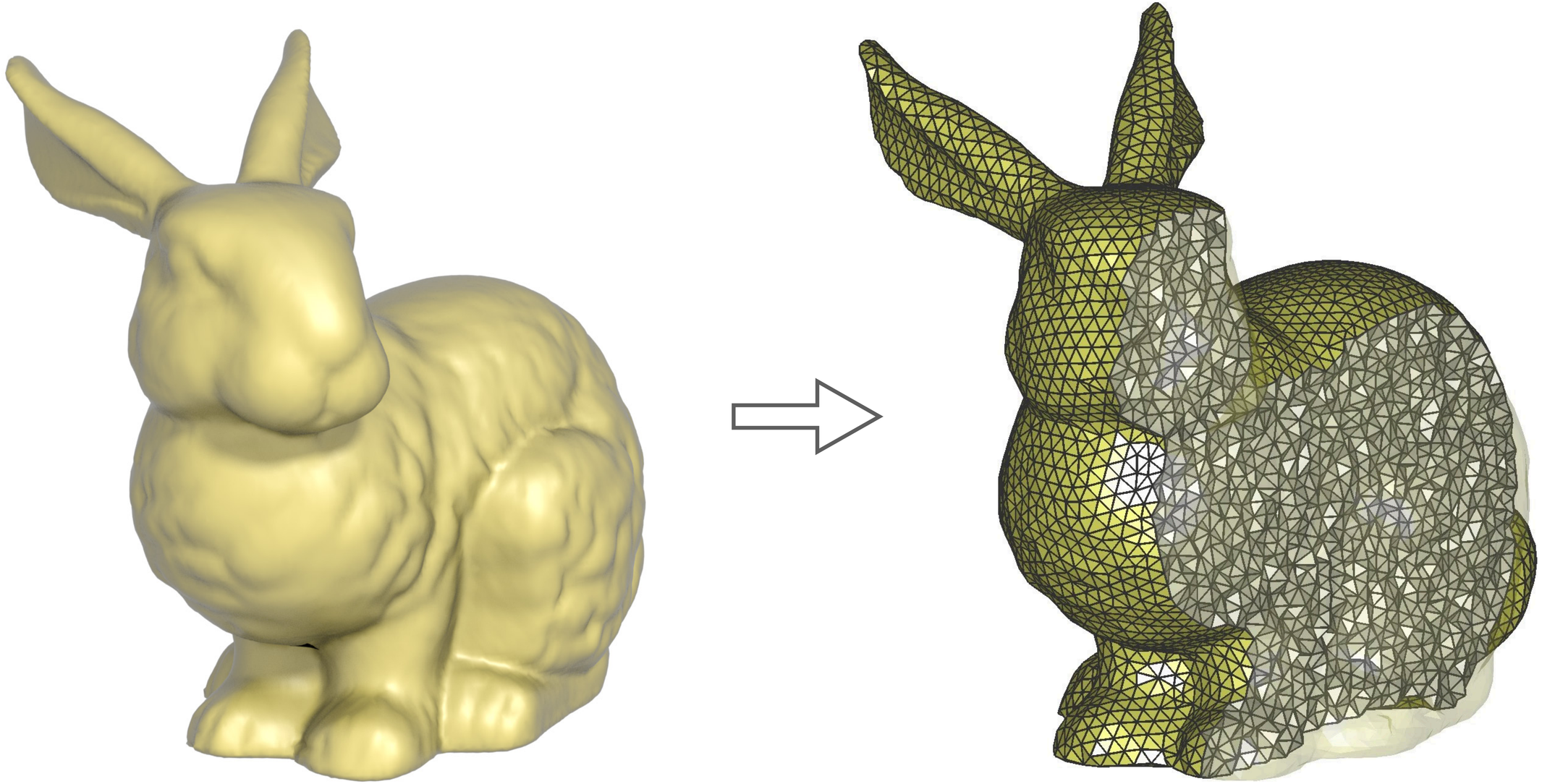


Tetrahedral Meshing in the Wild **[Hu et al. 2018]**



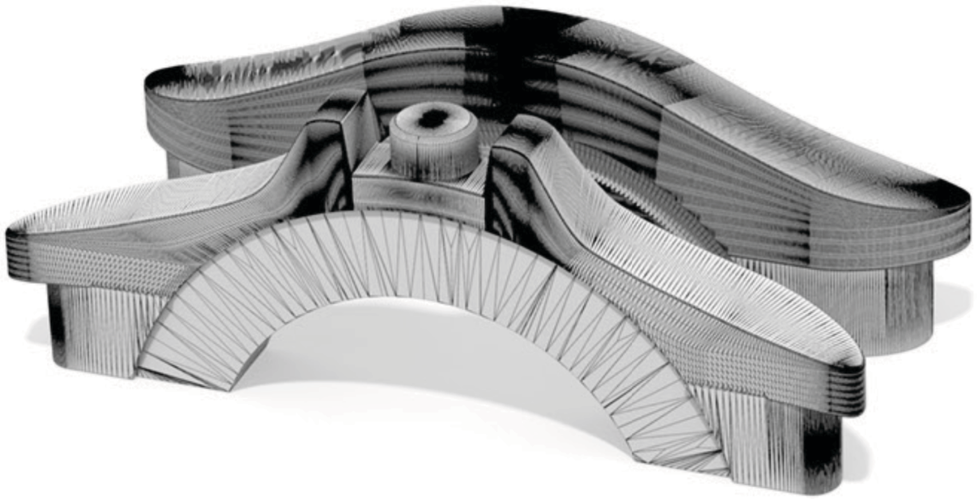
Fast Tetrahedral Meshing in the Wild **[Hu et al. 2020]**

Traditional PDE solvers require volumetric meshing

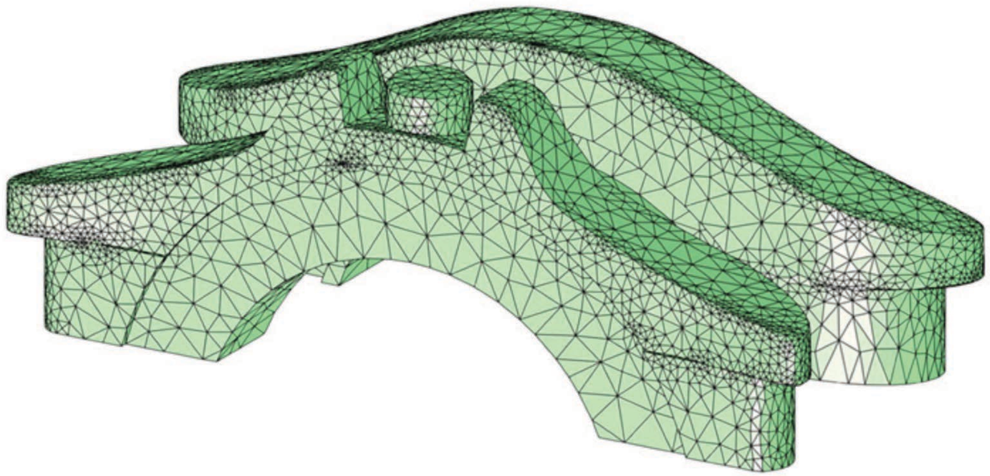


Robust meshing can be *wildly* unpredictable

Even very simple geometry can take **hours** to mesh:



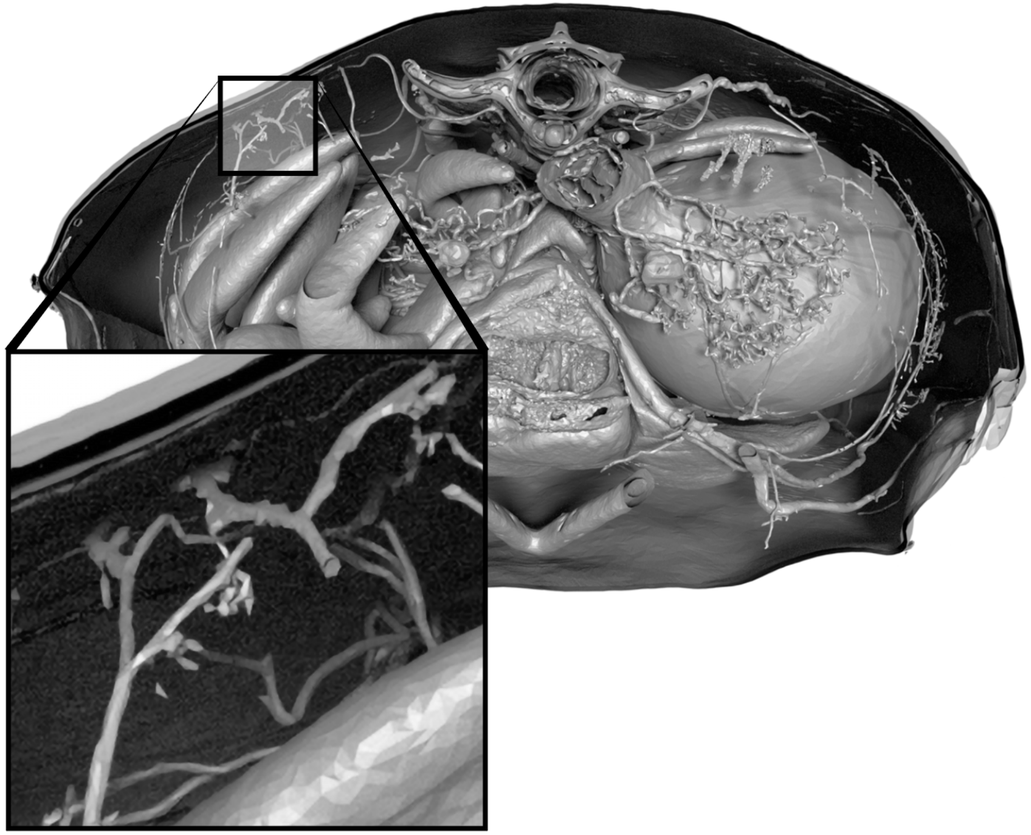
Input (Thingi10k #996816)



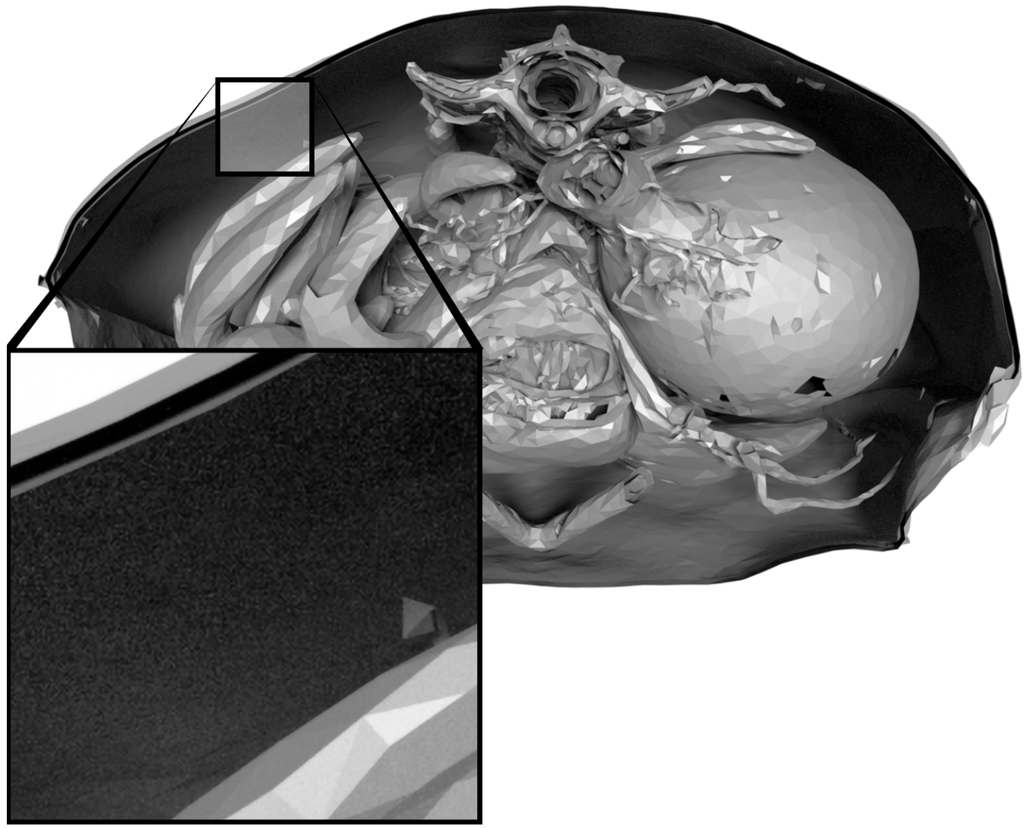
FastTetWild, **1 hour 25 minutes**

Hu et al, *"Fast Tetrahedral Meshing in the Wild"* (2020)

Even when meshing "succeeds", critical details can be lost

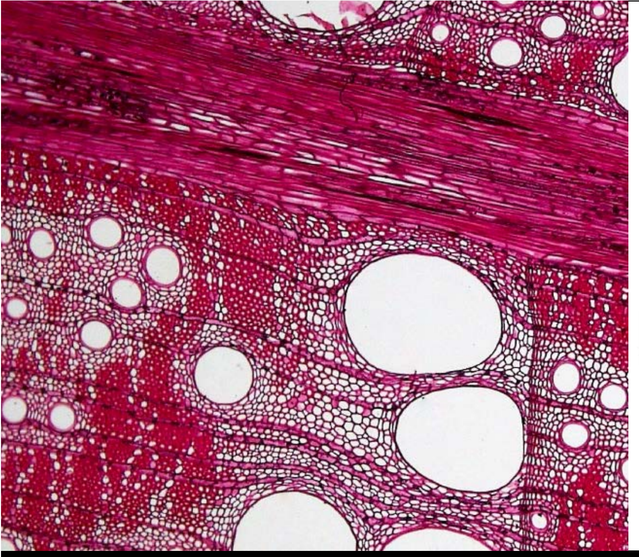


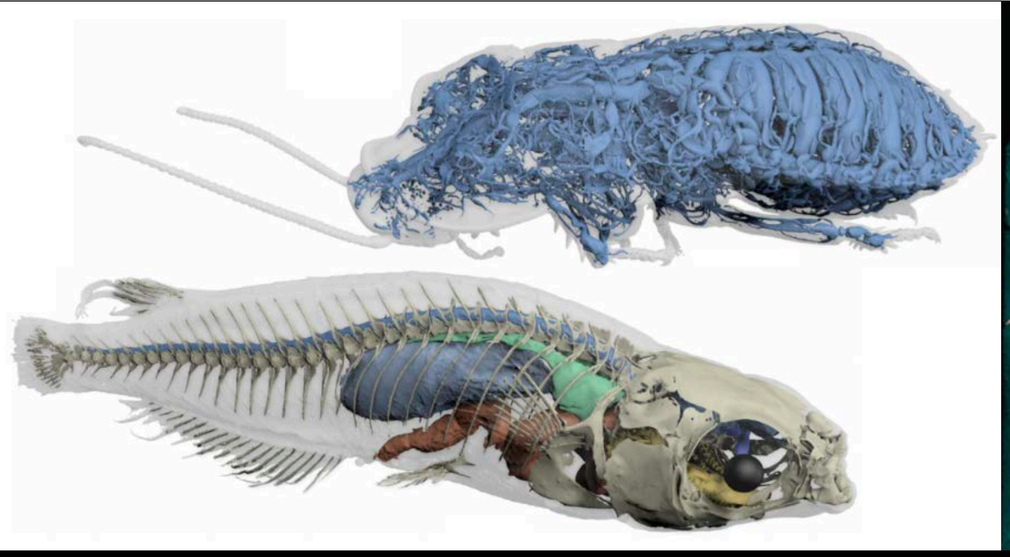
boundary mesh (*input*)

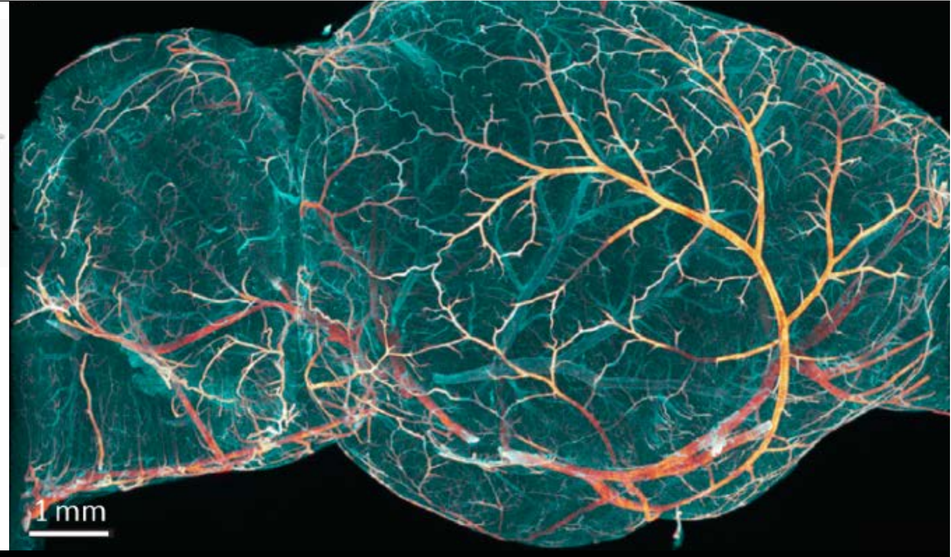


FastTetWild (*output boundary*)

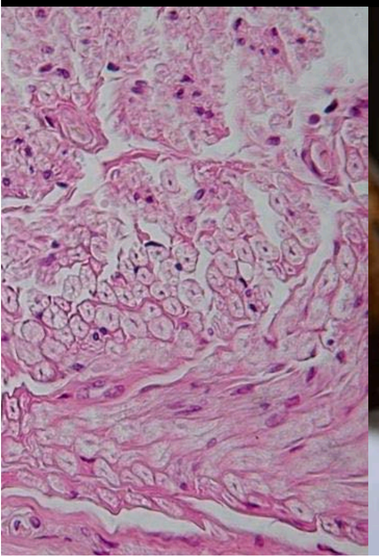
Hu et al, *"Fast Tetrahedral Meshing in the Wild"* (2020)

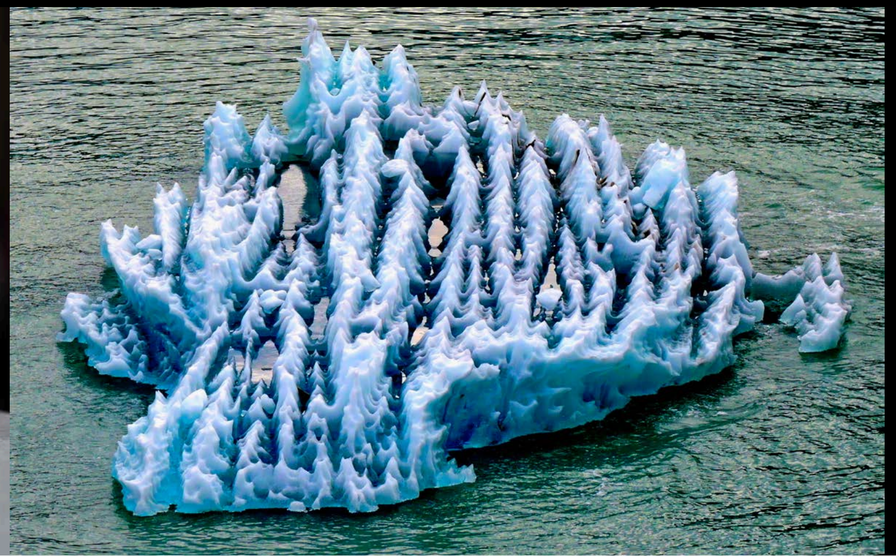






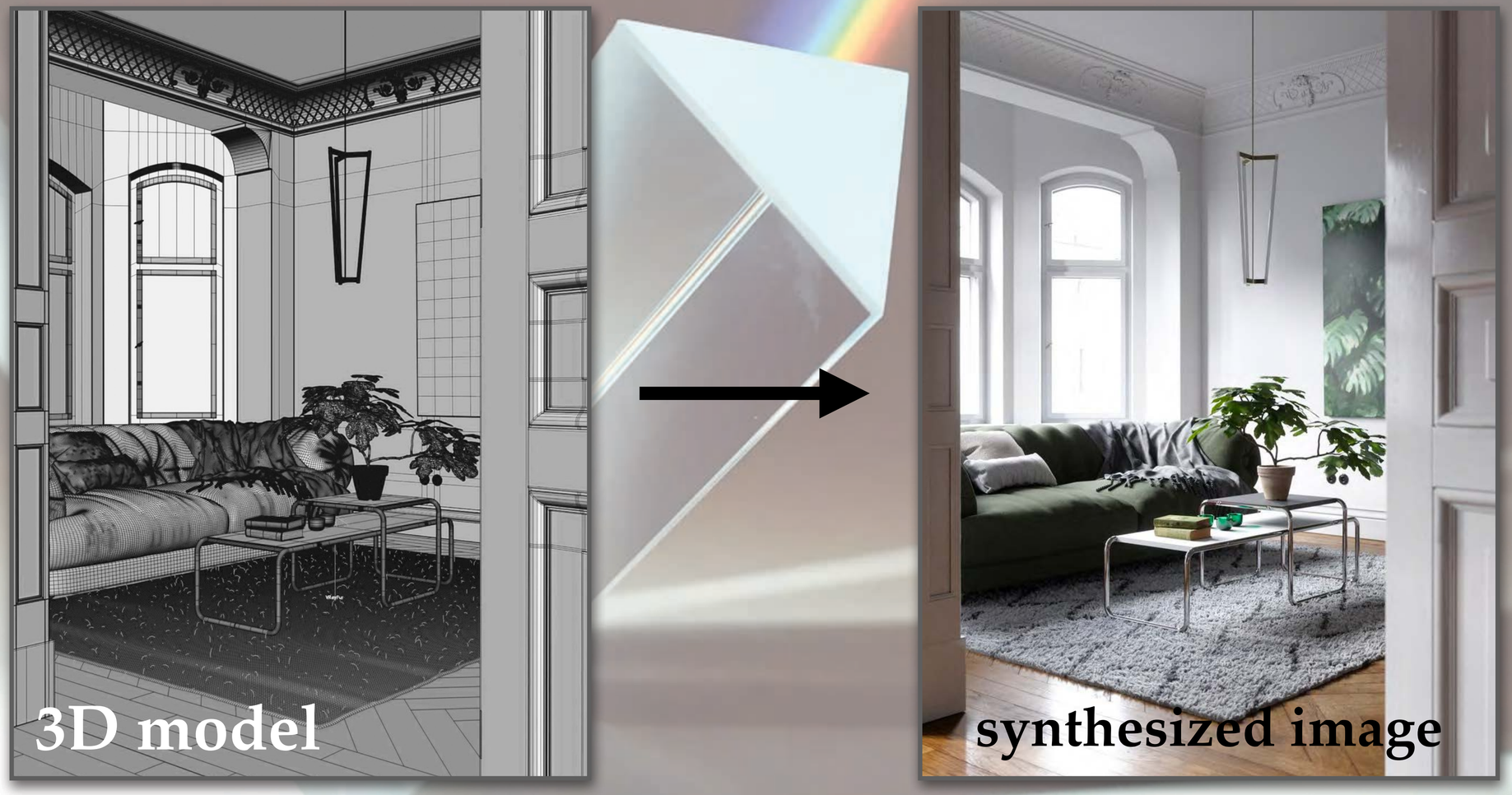
To faithfully simulate nature, we *must* be able to handle a much greater level of **geometric complexity**.





Photorealistic Image Generation

Problem: given a description of a 3D scene (geometry, materials, lights, camera), synthesize an image indistinguishable from a photograph.



Rendering: from Finite Elements to Monte Carlo

Early days of rendering: *finite element radiosity*

global & painful!

mesh scene

↓

setup large matrix

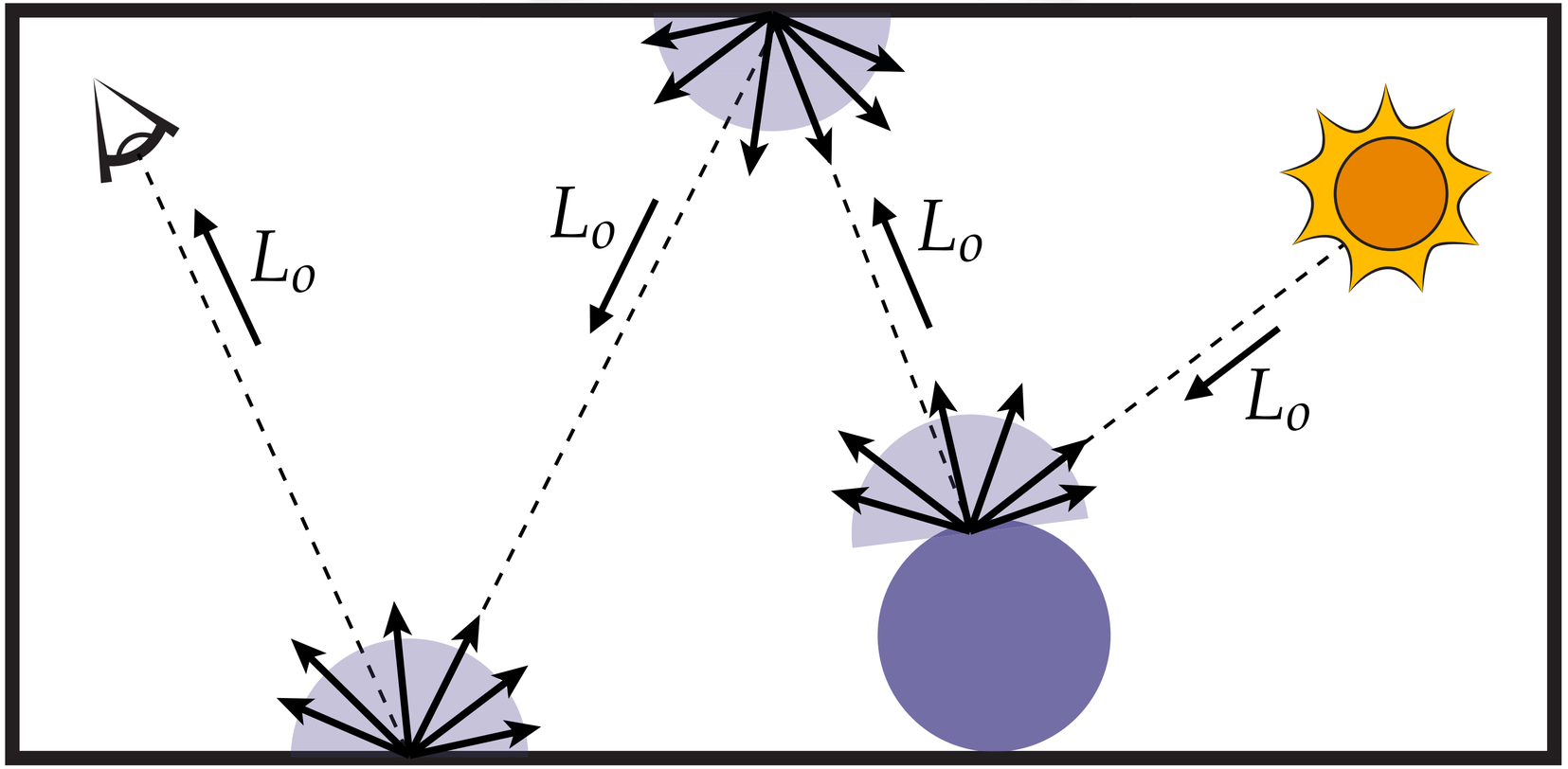
↓

perform global solve

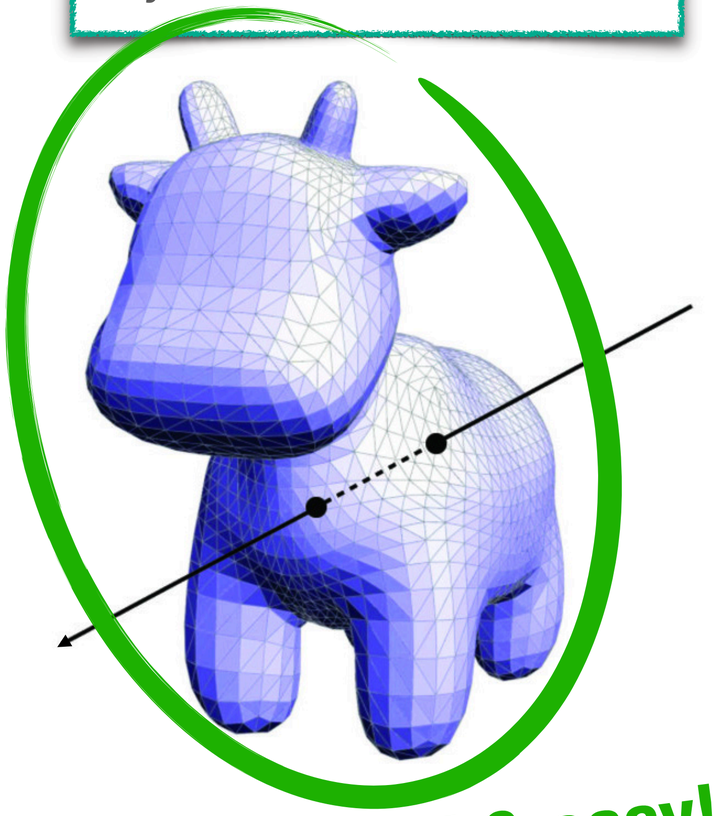
Rendering: from Finite Elements to Monte Carlo

Monte Carlo ray tracing avoids meshing **entirely**, via repeated random sampling

recursively shoot rays



ray-scene intersections



local & easy!

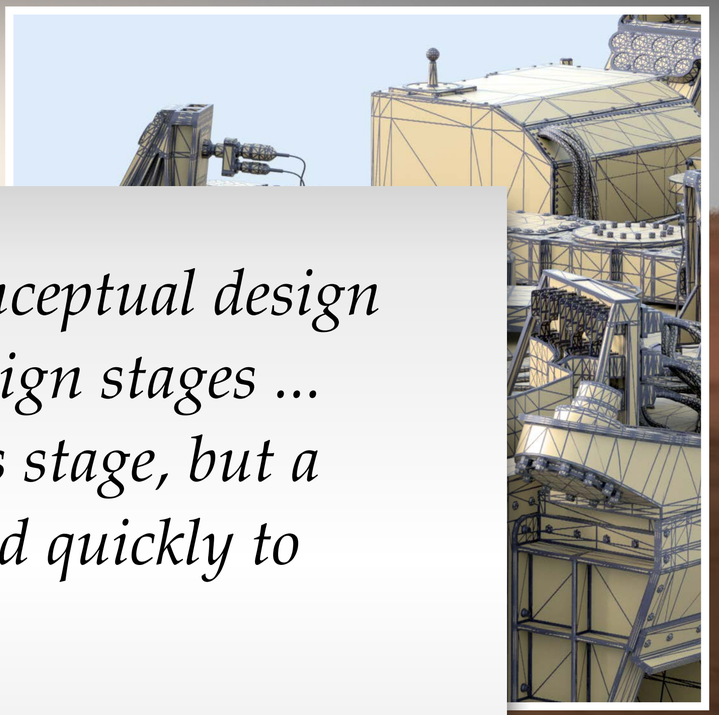
Monte Carlo methods

Evaluate an integral by averaging its integrand N times:

$$\int_{\Omega} f(x)\ dx \approx \frac{|\Omega|}{N} \sum_{i}^{N} f(X_i)$$

*Monte Carlo rendering can now handle **immense** geometric complexity*

a rendering of NASA's *Curiosity* Mars Rover

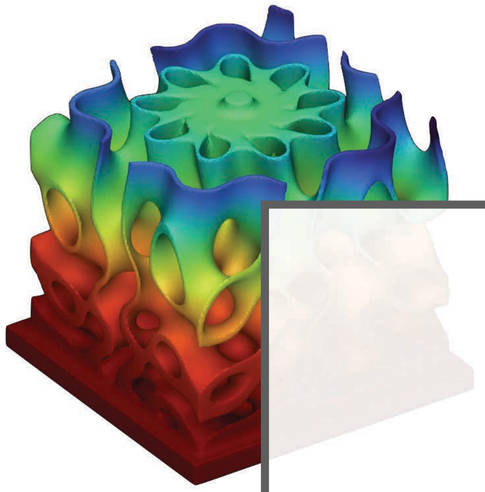


"Thermal modeling is required beginning at the project conceptual design stage and continuing through preliminary and detailed design stages ... simplified calculations and rules of thumb are useful at this stage, but a computer model provides the ability to evaluate and respond quickly to proposed system trade-offs."

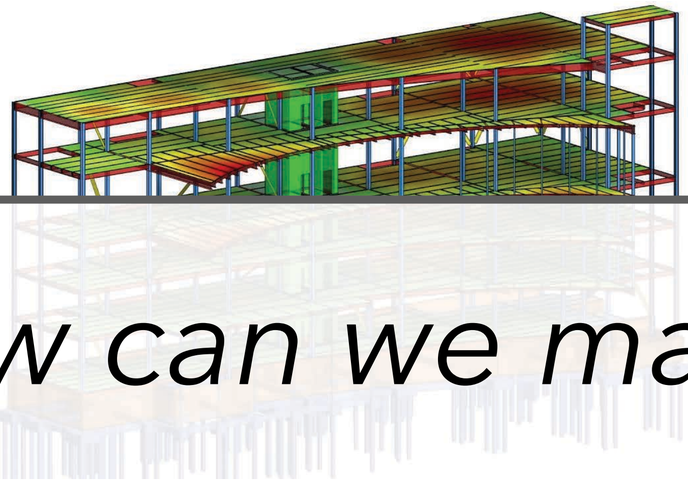
—NASA Guidelines for Thermal Analysis of Spacecraft Hardware

Rendering "just works," and gives immediate feedback, no matter what you throw at it.

Physics beyond light transport



thermal diffusion



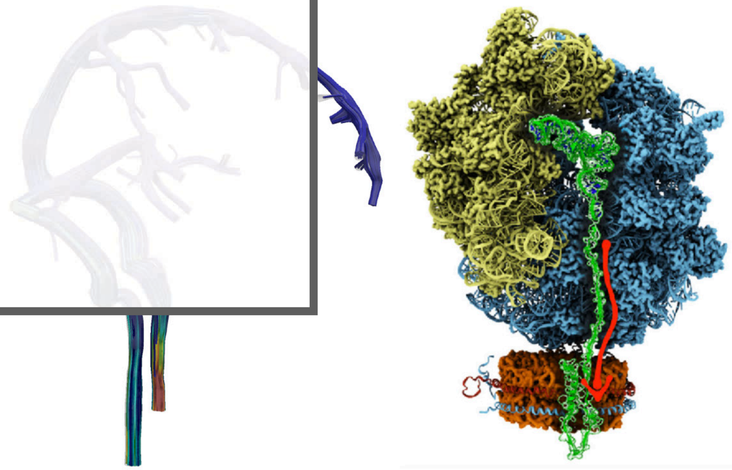
structural analysis

electrostatics

How can we make simulation more like rendering?

acoustic modeling

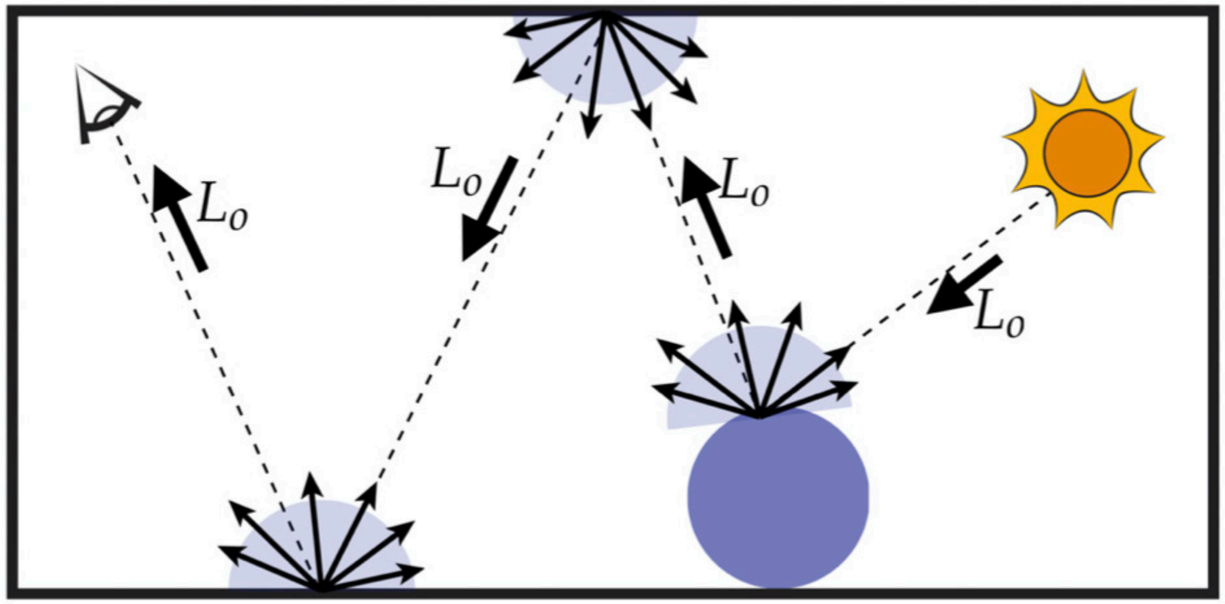
microfluidics



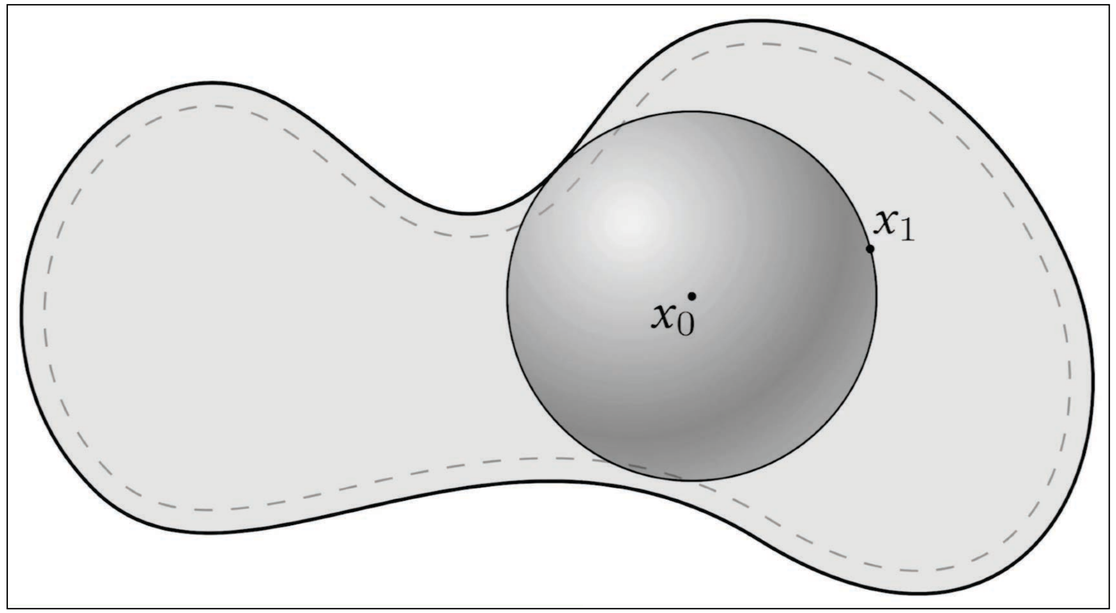
biophysics

From Ray Tracing to Random Walks

Recursive random walks for solving the Laplace equation $\Delta u = 0$

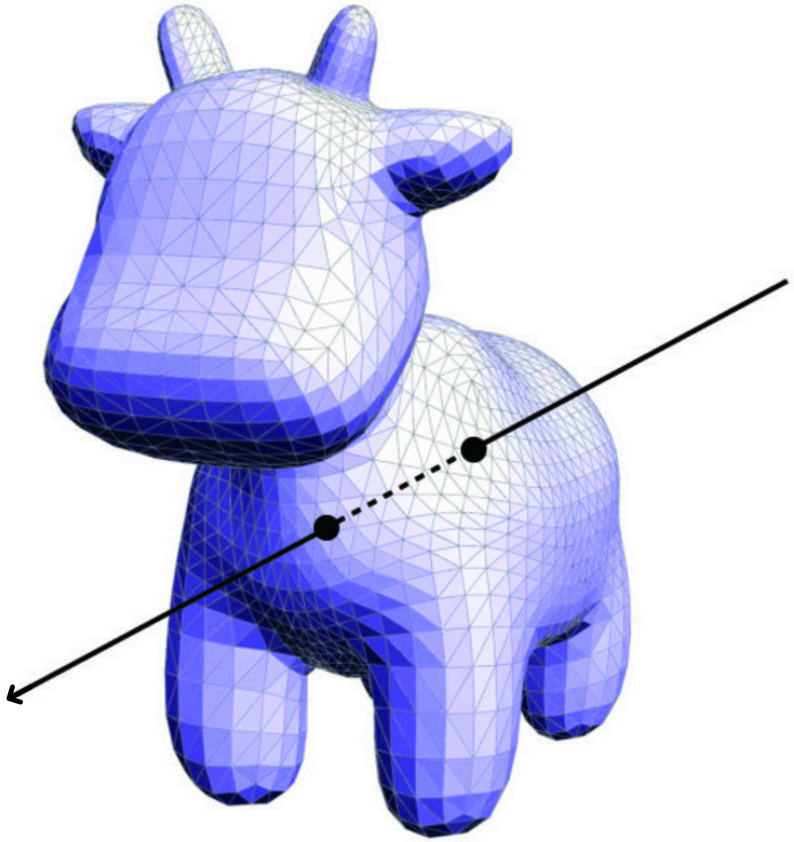


ray tracing

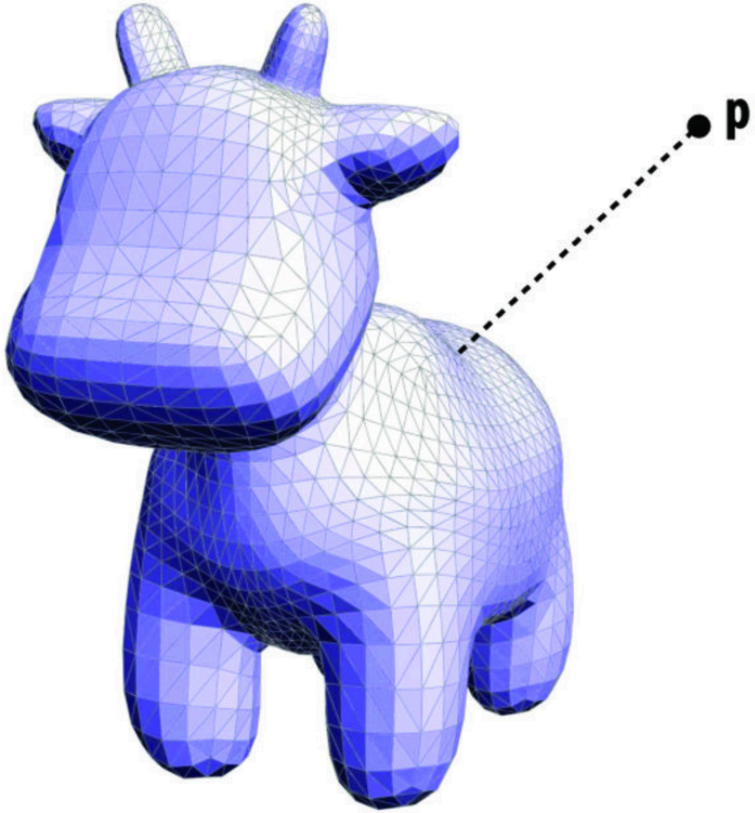


walk on spheres [Muller 1956]

From Ray Intersections to Closest Point Queries

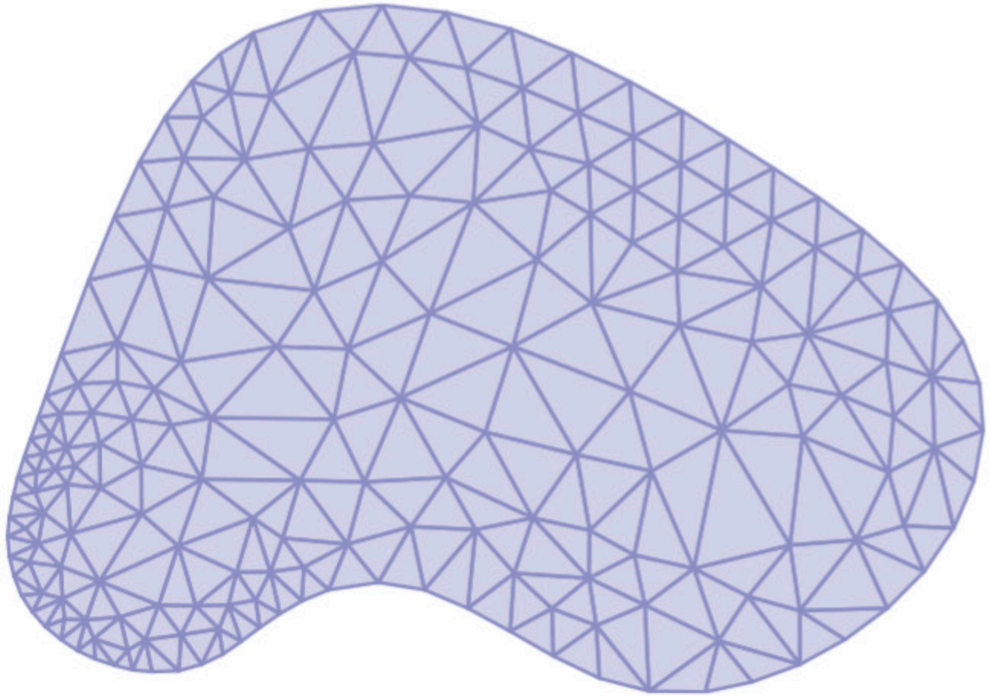


Ray Intersection Query

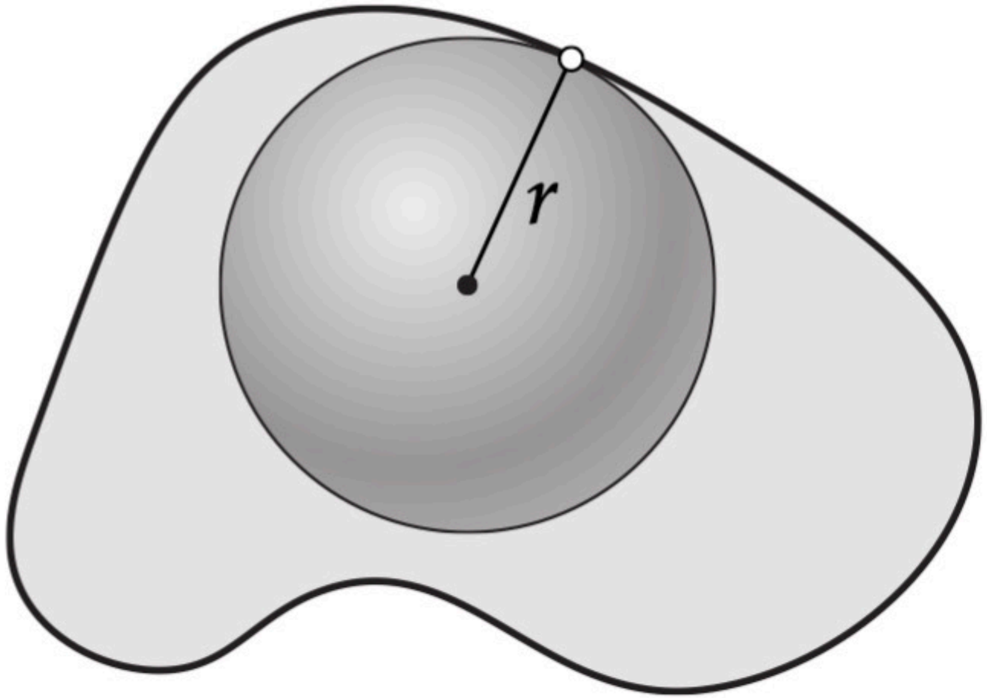


Closest Point Query

Meshing is hard…finding closest point is easy!

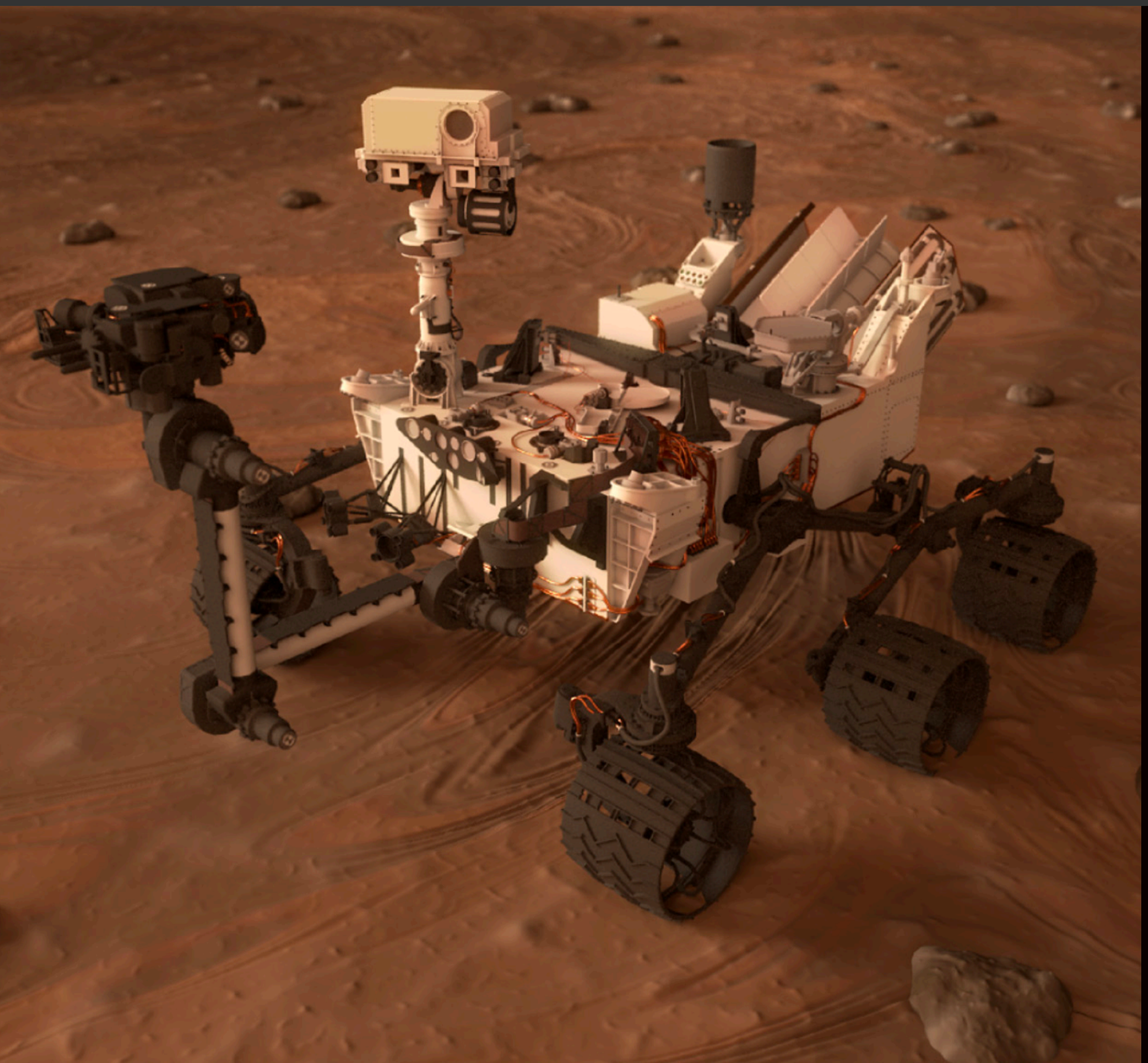


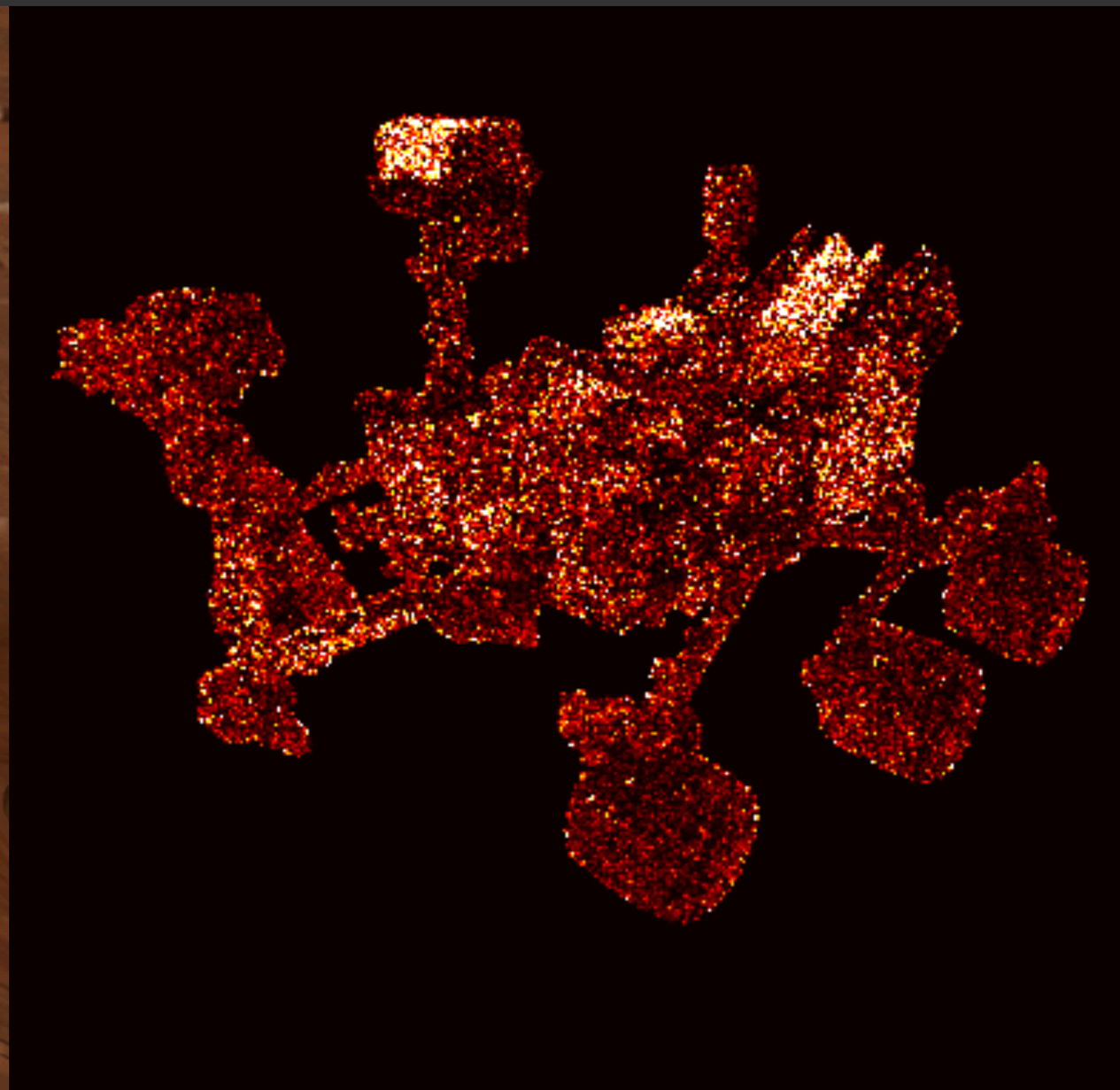
HARD



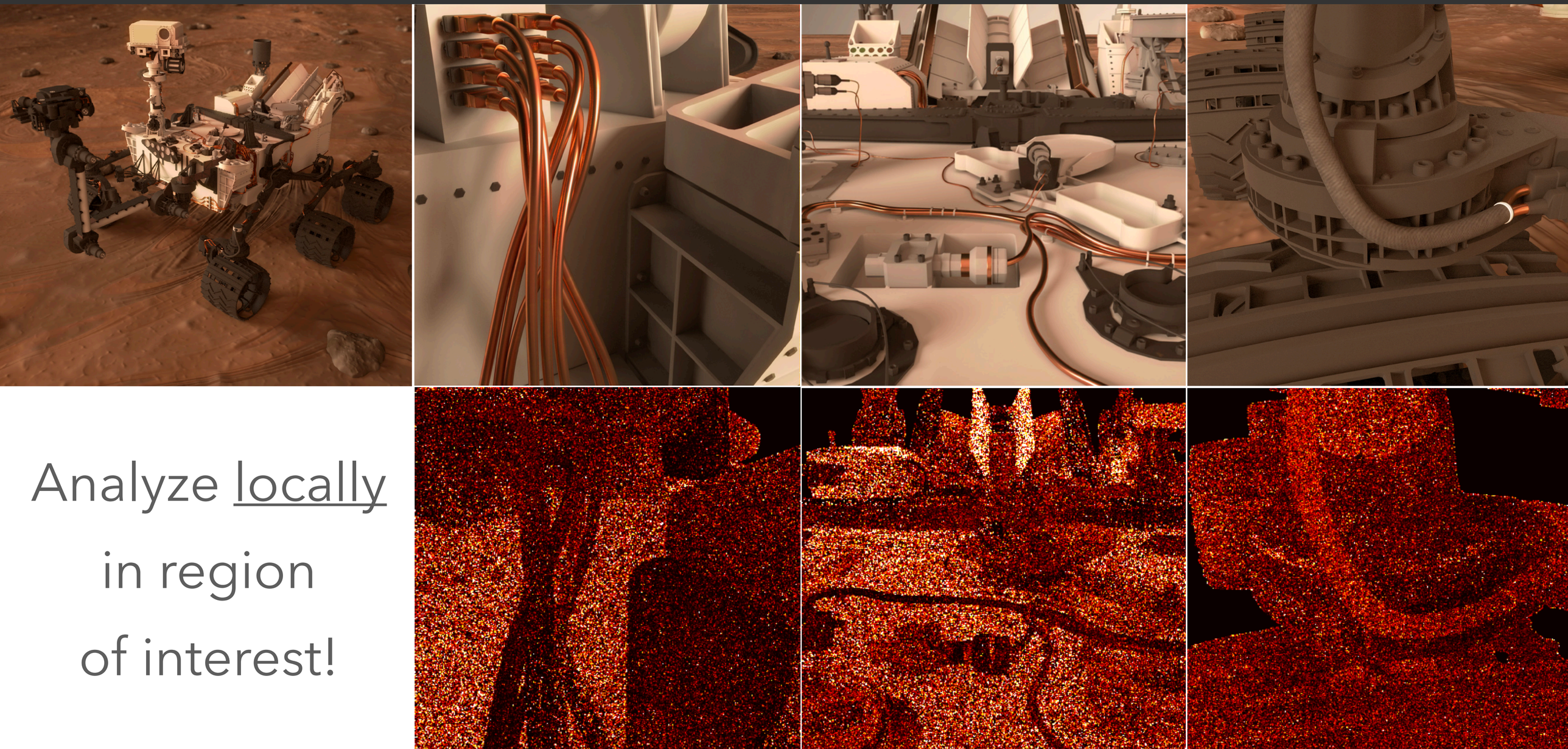
EASY

Thermal analysis of Curiosity Mars rover





Simulate only what you see!



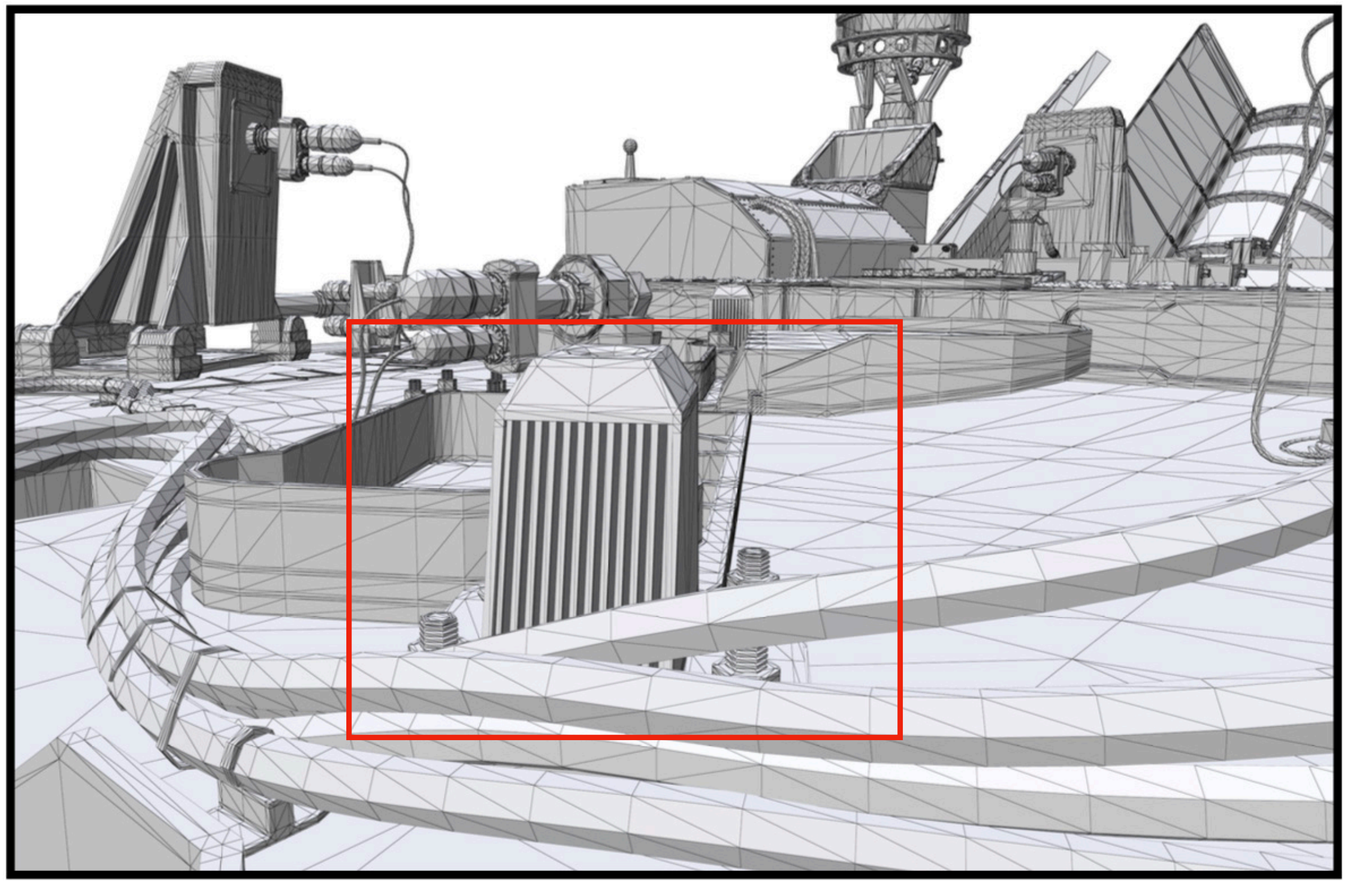
Analyze <u>locally</u>

in region

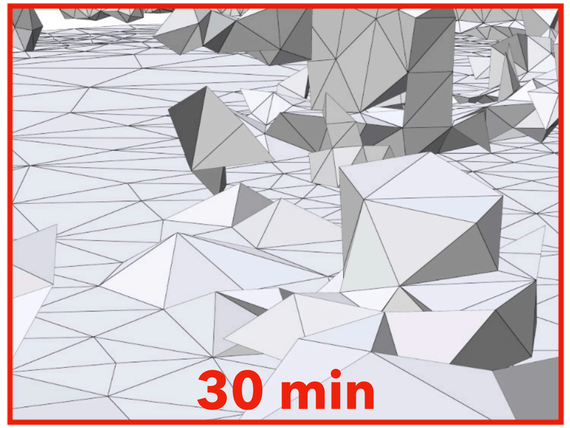
of interest!

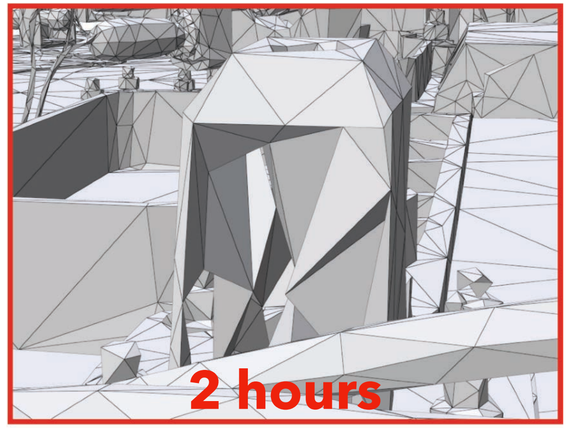
Thermal analysis is traditionally difficult in design phase

input boundary mesh



boundary of tetrahedral mesh

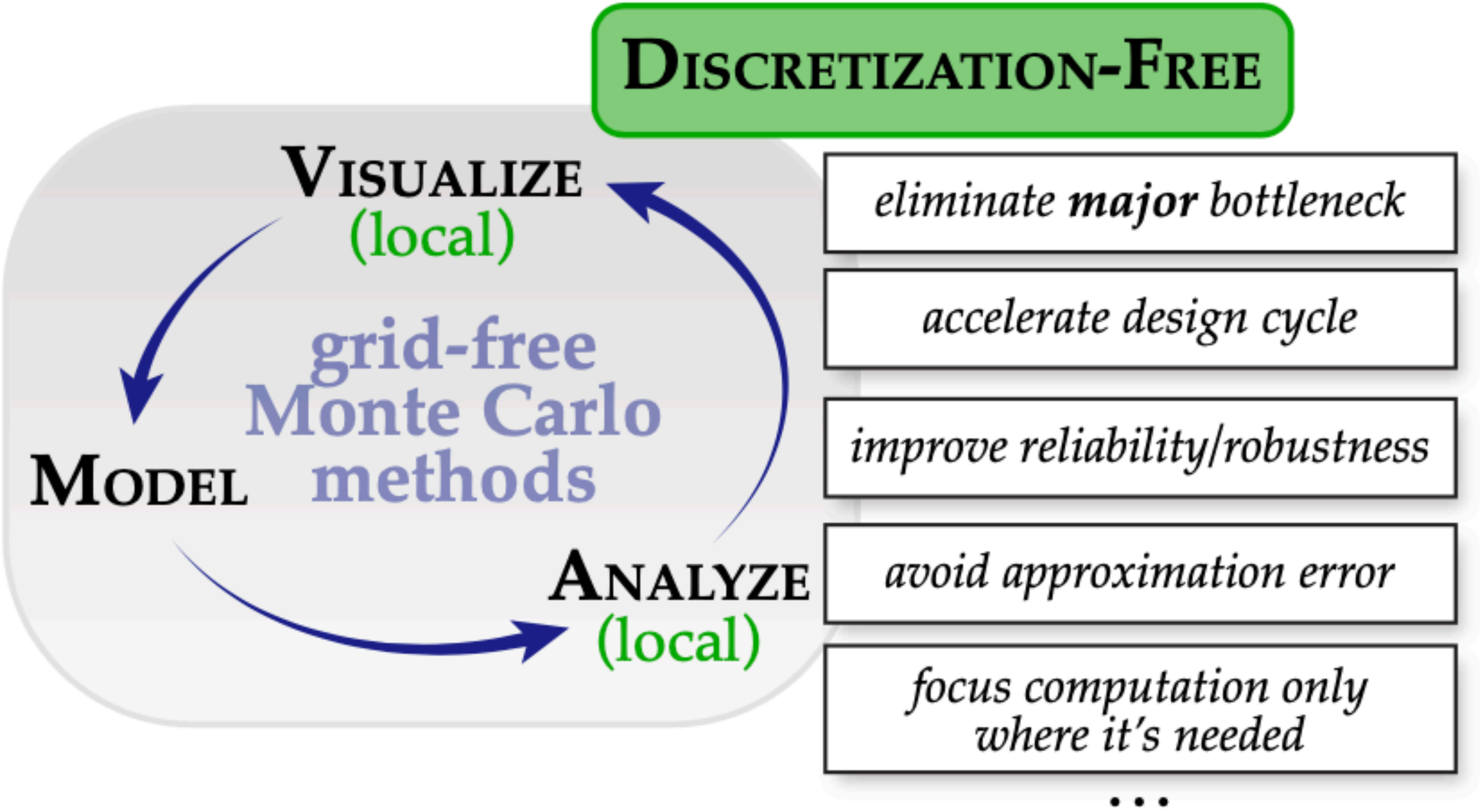




Out of memory

8 hours

Monte Carlo PDE solvers are discretization-free!

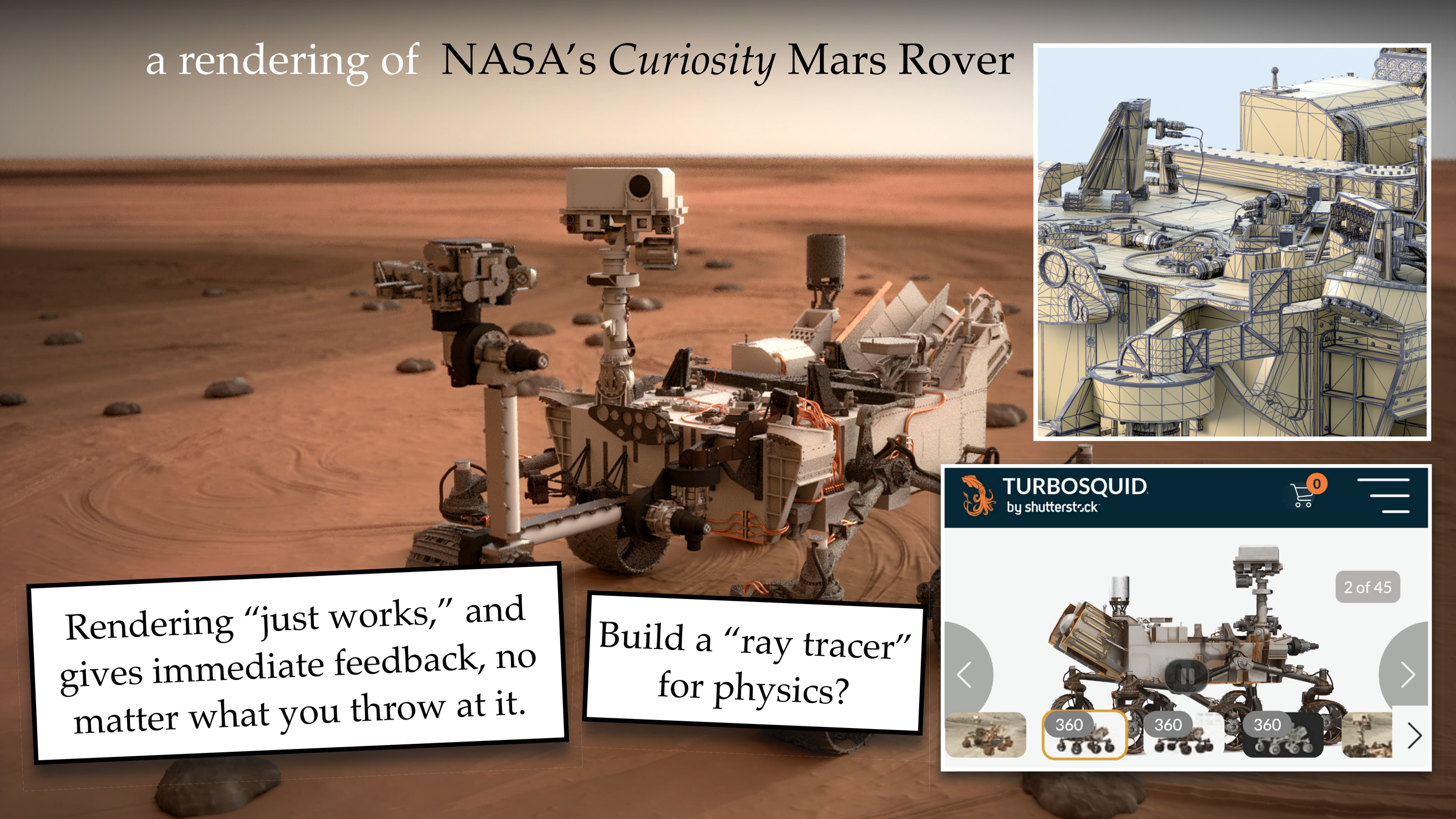


May never be able to solve certain PDEs w Monte Carlo…

Examples from:

Han et al, "A Hybrid Material Point Method for Frictional Contact with Diverse Materials" (2019)
Zhu et al, "Codimensional Non-Newtonian Fluids" (2015)

a rendering of NASA's *Curiosity* Mars Rover



Rendering "just works," and gives immediate feedback, no matter what you throw at it.

Build a "ray tracer" for physics?

Course overview

Parts 1–2: Basics of Monte Carlo & WoS for Laplace eq (Bailey)

– **key concepts:** estimation of integrals, sample generation, bias

Parts 3–4: WoS for Poisson eq (Bailey) & Neumann boundary conditions (Rohan)

– **key concept:** importance sampling

Part 5: WoS as simulation of Brownian motion (Rohan)

Part 6: Variance reduction (Bailey)

Part 7: Evaluation, recent work & future directions (Bailey)

PART 1:
Basics of Monte Carlo

Thanks in advance!

Monte Carlo Methods and Applications

CMU 21-387 | 15-327 | 15-627 | 15-860 FALL 2024

geometry.cs.cmu.edu/montecarlo

Home — Course Info — Schedule — Assignments — Resources — Course Policies

Instructors: Keenan Crane (CSD/RI) and Gautam Iyer (MSC)

Location: Scott Hall (SH) 105

Date and time: Tuesday / Thursday, 12:30am–1:50pm

Units: 9 (3 in-class/6 outside)

Many slides based on
Keenan Crane &
Gautam Iyer's
Monte Carlo course at CMU,
thanks Keenan & Gautam!!

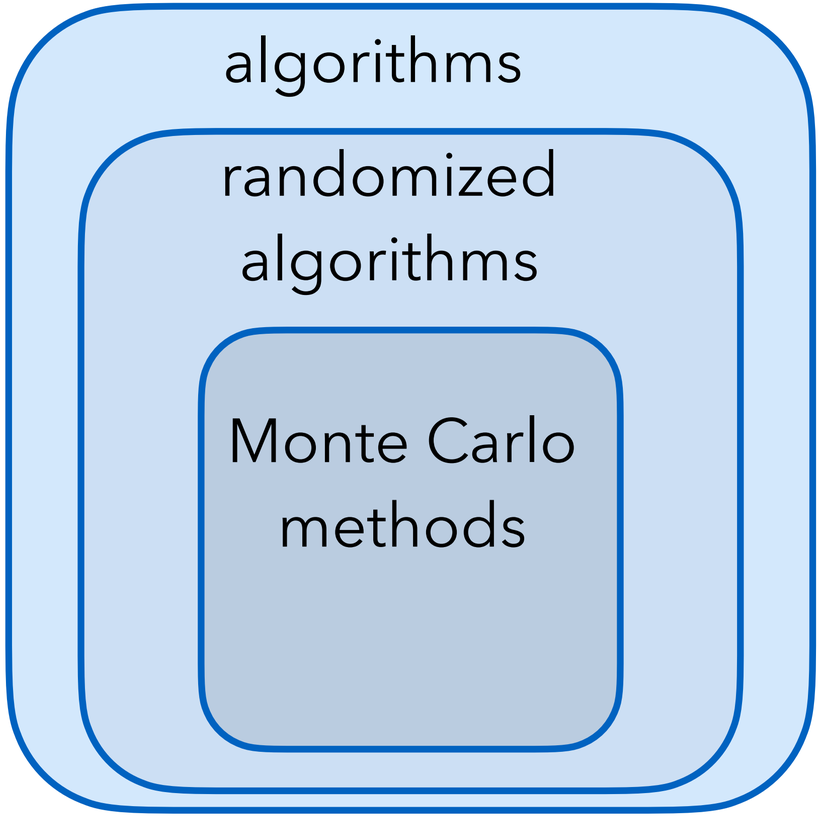
What are Monte Carlo methods?

- Broadly, **Monte Carlo methods** are algorithms that use *repeated random sampling* to obtain approximate solutions to difficult computational problems

- simulation, **integration**, optimization, sampling

- Not all *randomized algorithms are Monte Carlo methods*.

- E.g., *Las Vegas algorithms* use repeated random trials to get an *exact solution*, but with nondeterministic runtime (e.g. randomized quick sort)

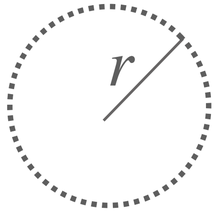


Flavor of Monte Carlo: computing area of a cookie

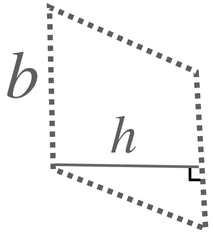
motivation: selling custom shaped cookies, want to set price based on cookie size

challenge: how do you compute the area of an arbitrary cookie?

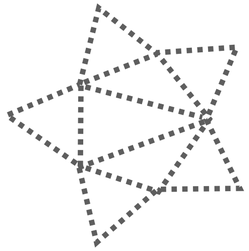
naive: compute analytically or decompose into simple shapes



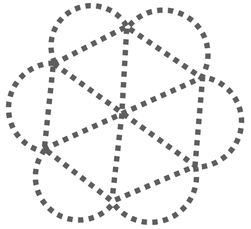
$$\pi \cdot r^2$$



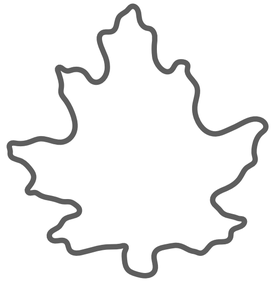
$$b \cdot h$$



$$\sum_{i=1}^{8} \underbrace{\frac{b_i \cdot h_i}{2}}_{\text{triangle area}}$$



$$\sum_{i=1}^{6} \underbrace{\frac{b_i \cdot h_i}{2}}_{\text{triangle area}} + \sum_{j=1}^{6} \underbrace{R_j^2 cos^{-1}\left(\frac{R_j - h_j}{R_j}\right) - (R_j - h_j) \cdot \sqrt{2 \cdot R_j \cdot h_j - h_j^2}}_{\text{circular segment area}}$$



?

area becomes less trivial to compute

Flavor of Monte Carlo: computing area of a cookie

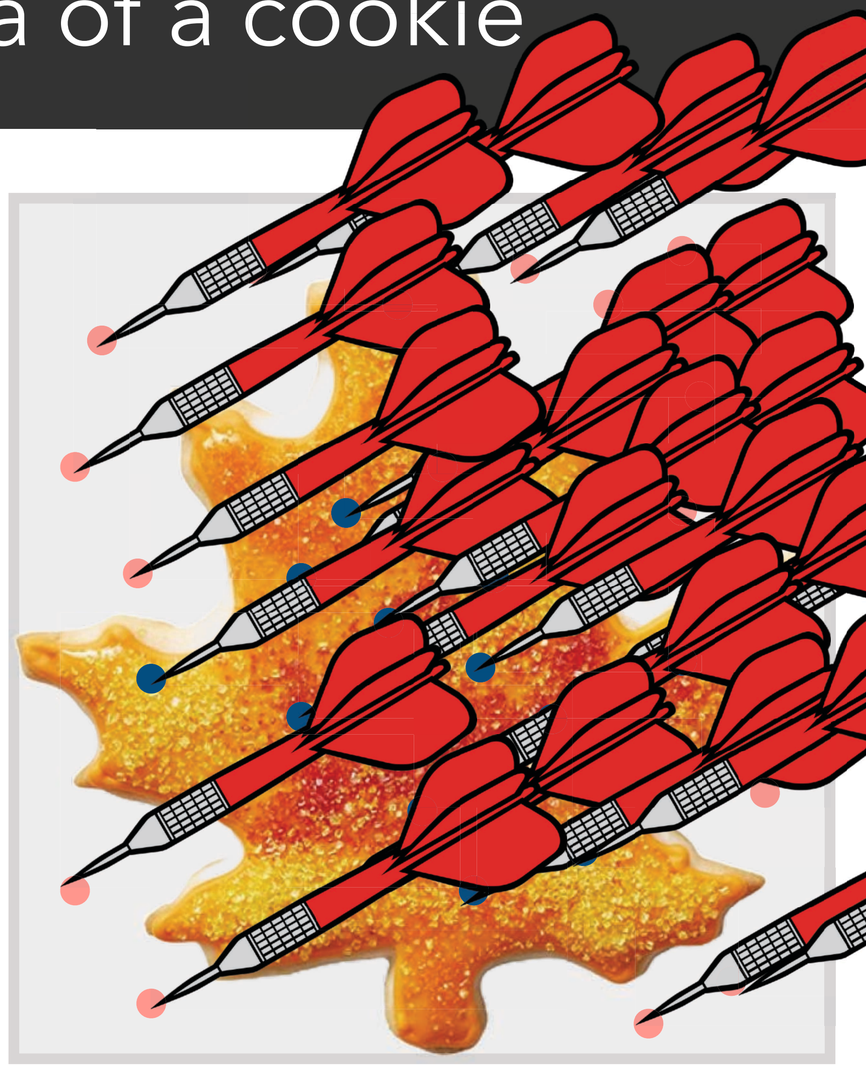
Monte Carlo approach to computing area:

1. throw N random darts at plate containing the cookie
2. compute proportion of darts that hit cookie
3. multiply proportion by area of plate

$$\textbf{cookie area} \approx \frac{\textbf{\# darts hit cookie}}{\textbf{\# total darts thrown}} \bullet \textbf{plate area}$$

now we can easily handle complex shapes:

bounding plate



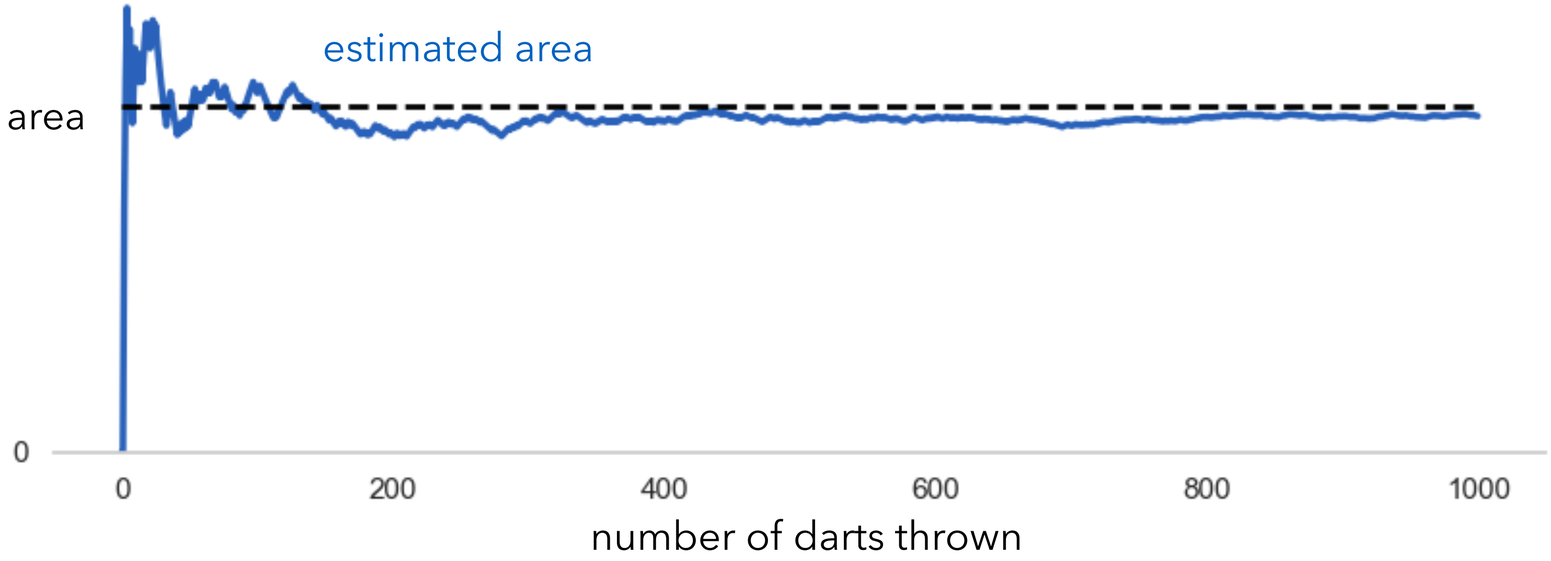
dart hits cookie

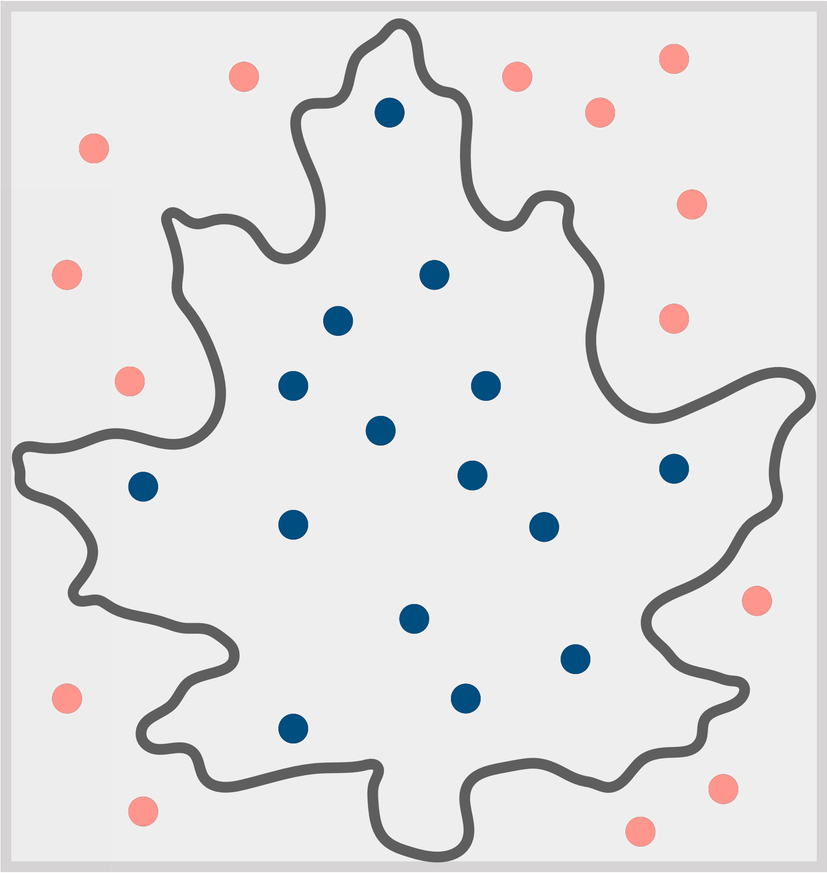
dart misses cookie

Flavor of Monte Carlo: computing area of a cookie

Already have some important takeaways:

- algorithms can be extremely simple
- solution is approximate, but correct eventually
- easy to parallelize: average independent trials





Solving problems with integration

- Many problems can be reframed as <u>integration problems</u>: computing area, light transport, thermal conduction, maximum likelihood estimation, etc.

write down problem

Find u such that

$$\Delta u = f$$

rewrite as integral

$$u(x) = \int_{\mathbb{R}^N} G(x, y) f(y)\ dy$$

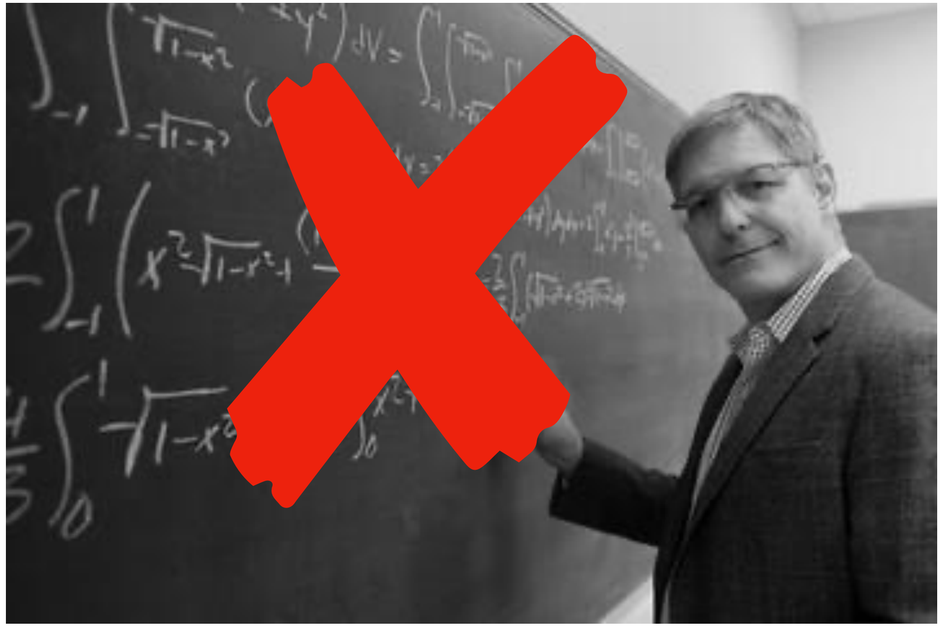
evaluate integral

$$u(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{G(x, y_i) f(y_i)}{p(y_i)}$$

- Primarily focus on using integration to solve **partial differential equations** with walk on spheres

Most integrals do not admit closed form solutions

- Forgot your integration rules? No problem!
- Analytic integration is often not a viable option
- Even simple looking expressions may have no integral expression in terms of elementary functions (Liouville theorem).



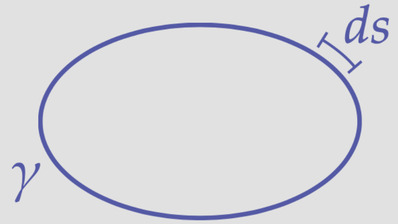
$$\int_0^1 \sin(x)\, dx$$

Sure.

$$\int_0^1 \frac{\sin(x)}{x}\, dx$$

Nope.

$$\text{length} = \int_\gamma ds$$



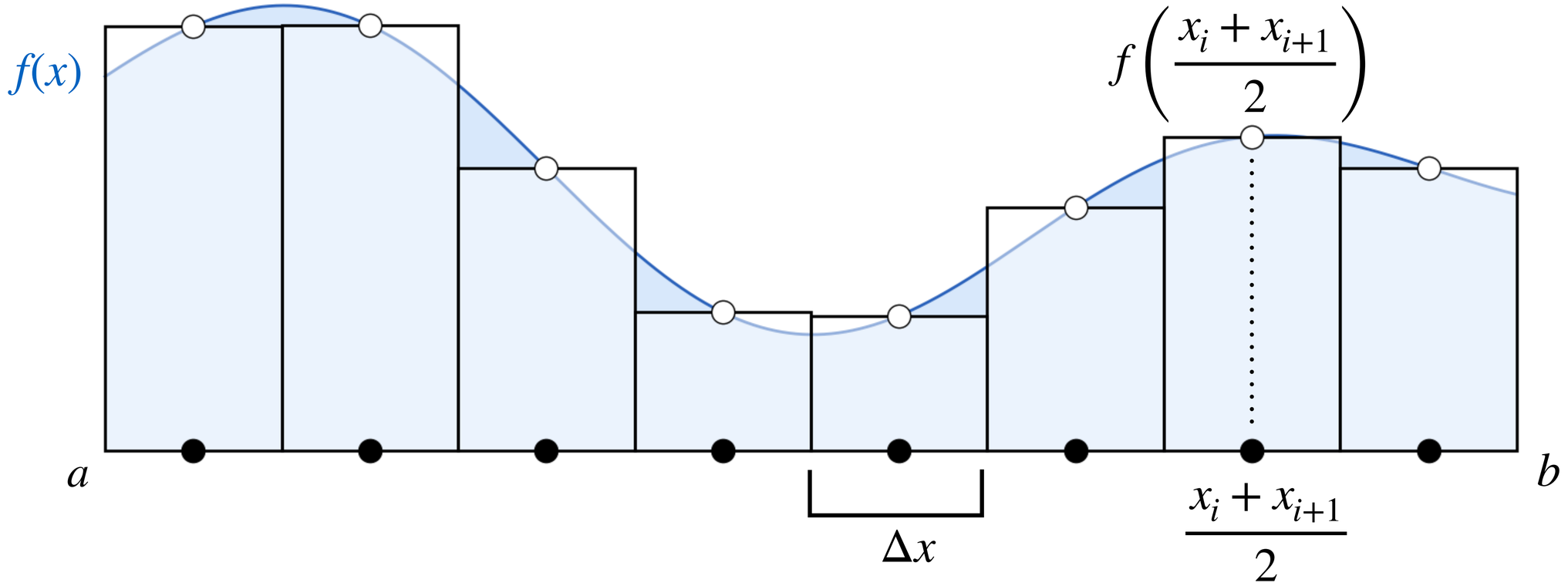
Think again.

Numerical quadrature

- Divide the integral up into discrete intervals we can approximate as constant functions
- Deterministic and straightforward to evaluate in this 1D example, what's the catch?

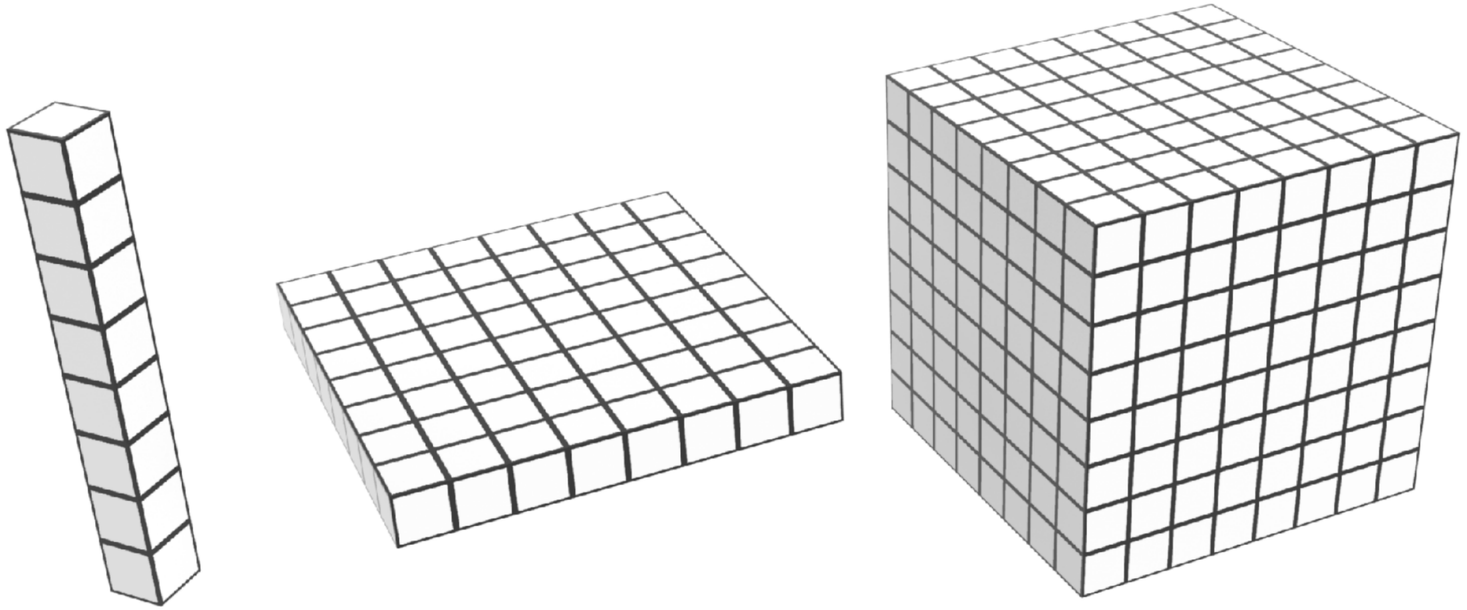
example:
(Midpoint rule)

$$I = \int_a^b f(x)\ \mathrm{d}x \approx \sum_{i=1}^{n} f\left(\frac{x_{i+1} + x_i}{2}\right) \Delta x$$



Curse of dimensionality

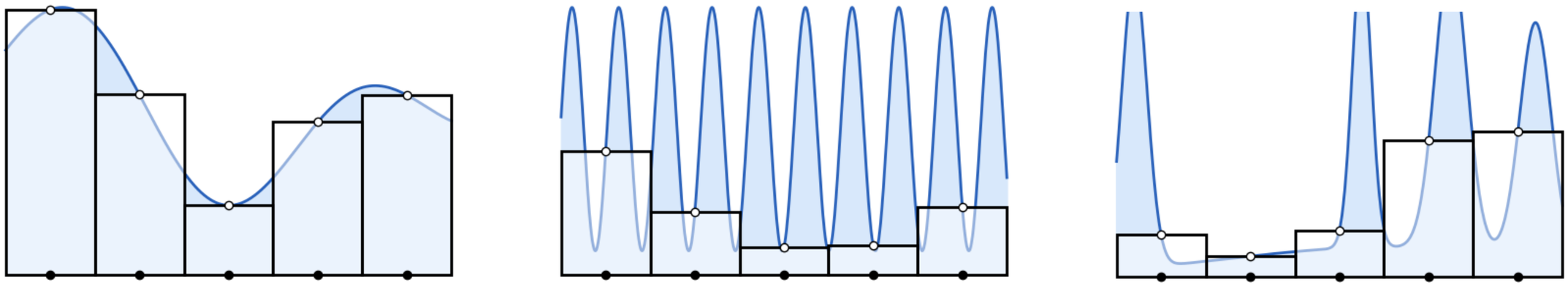
- In higher dimension, we divide the domain up into higher order voxels
- Cost grows exponentially as dimension increases!



$O(n)$ $O(n^2)$ $O(n^3)$ $\cdots$ $O(n^k)$

Aliasing

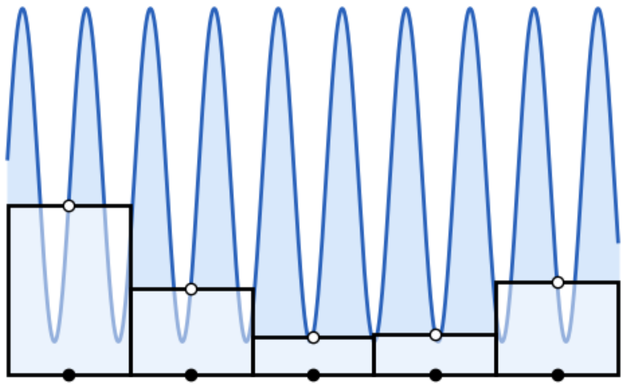
Confounding issue: often don't have a nice smooth integrand



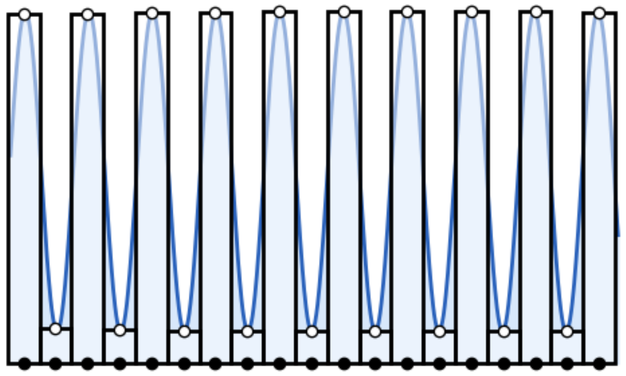
smooth **oscillatory** **spiky**

Aliasing – Nyquist-Shannon

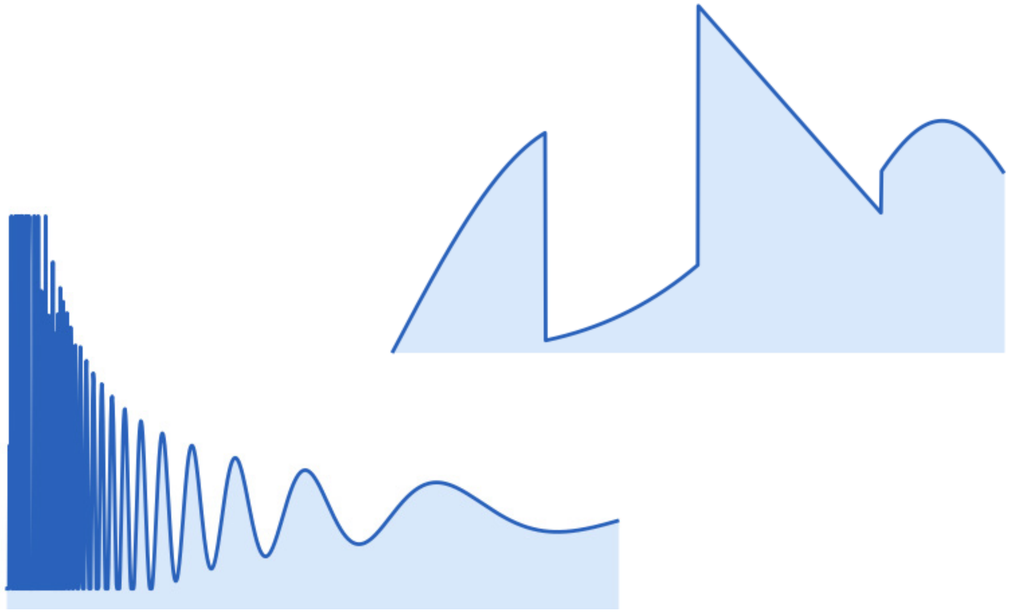
Nyquist-Shannon sampling theorem says we're basically ok as long as sample rate is adapted to the highest frequency…



oscillatory



more samples



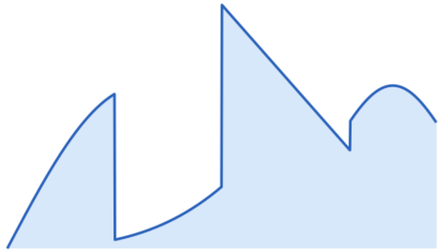
out of luck

…but not every function has a highest frequency

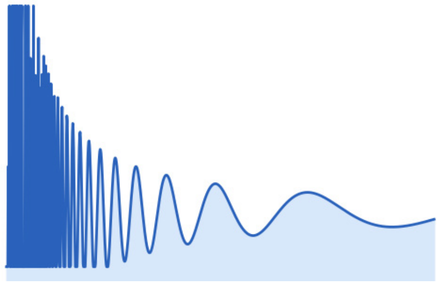
Aliasing in real functions

These aren't just pathological cases, these features are commonly found in nature

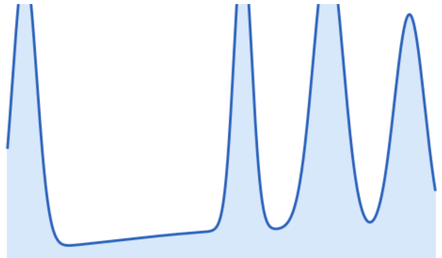
discontinuities
bridge masks background



high frequency
entire city skyline with buildings



spikes
Lighting from cars on bridge



Inspiration from Monte Carlo rendering

These seemingly hopeless integration problems can become tractable with the right Monte Carlo methods

real photograph

rendered scene from "Big Hero 6"

Monte Carlo to the rescue

Central idea: replace deterministic sample points with random ones

volume of domain

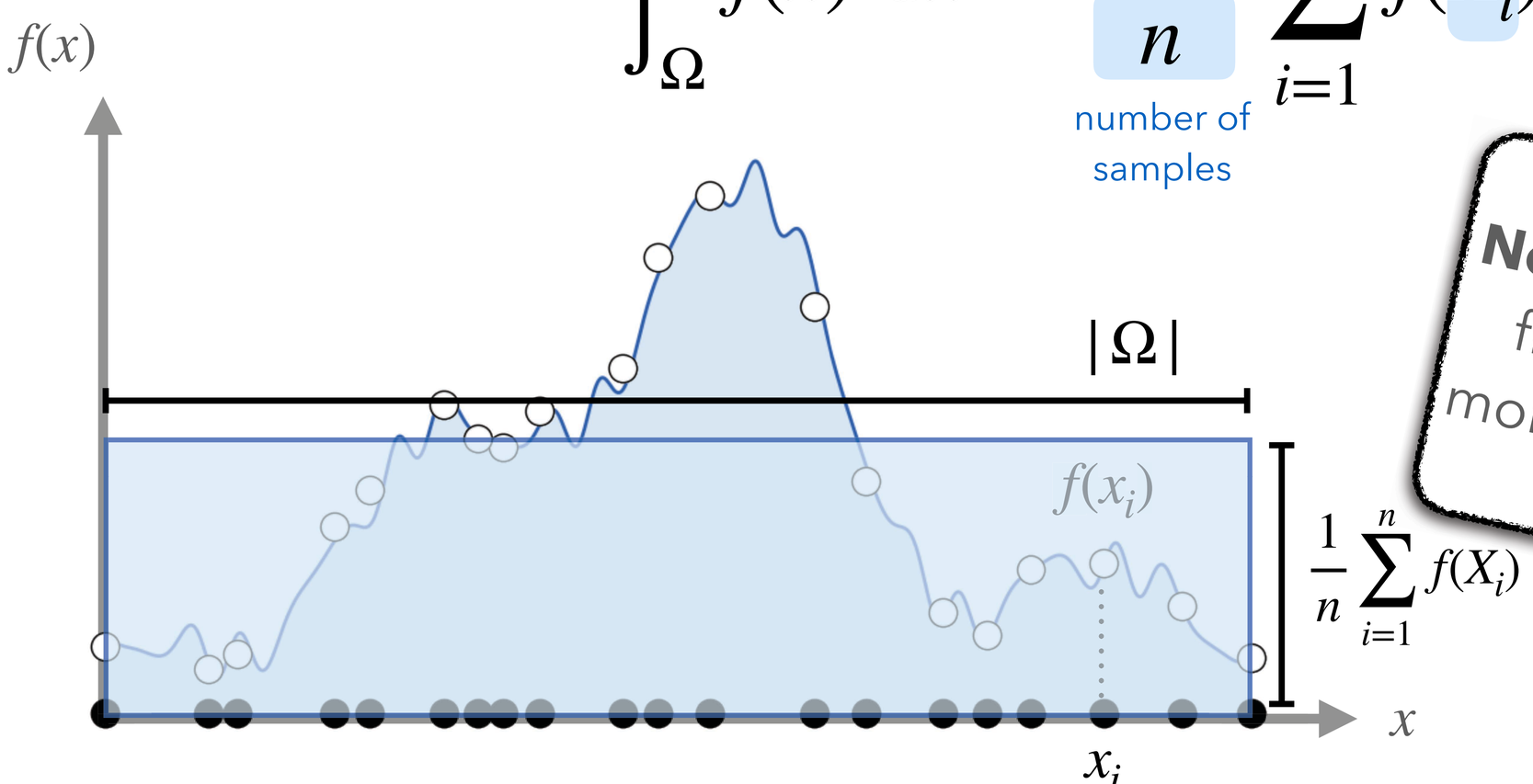
random sample point

$$\int_\Omega f(x)\ \mathrm{d}x \approx \frac{|\Omega|}{n} \sum_{i=1}^{n} f(X_i)$$

number of samples

$$X_i \sim \mathcal{U}_\Omega$$

uniformly distributed points

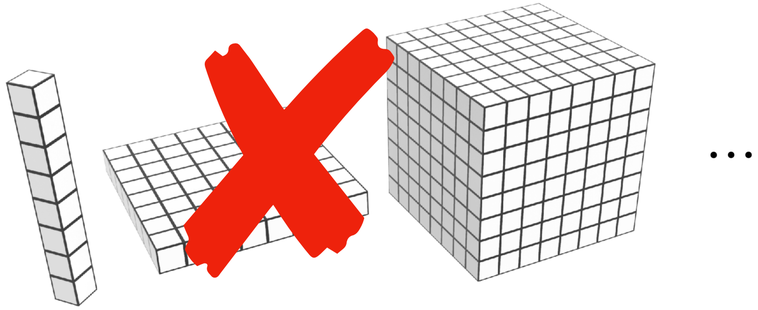


Note: number of samples is not fixed, can progressively add more samples to refine estimate

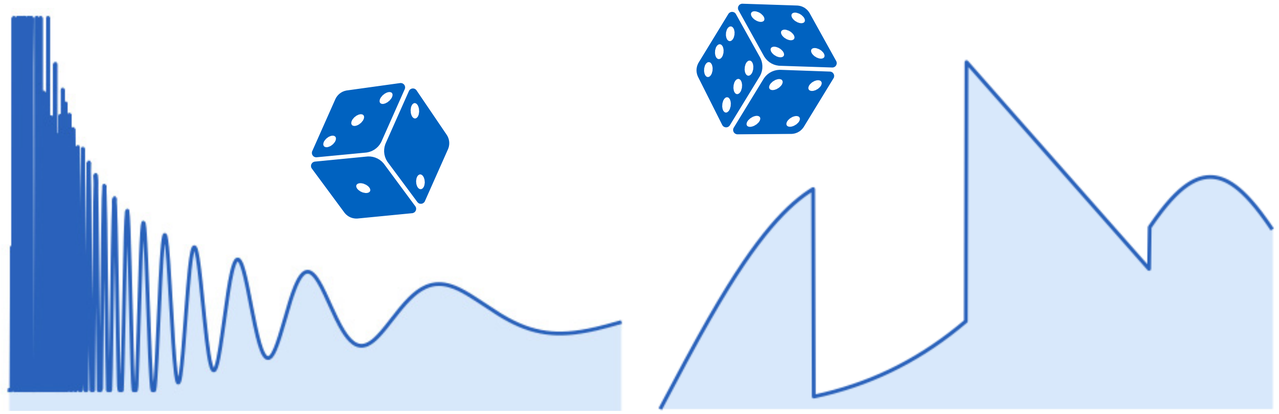
Monte Carlo to the rescue

Solves both of the problems with quadrature

Chose the amount of computation to do since rate of **convergence does not depend on dimension**



Always some chance of sampling high frequency features—**no "fixed" highest frequency**



How do we know Monte Carlo integration is "correct"?

A Monte Carlo estimator is a **random variable** since it is a function of random samples.

$$I := \int_\Omega f(x)\ \mathrm{d}x \qquad \hat{I}_n = \frac{|\Omega|}{n}\sum_{i=1}^{n} f(X_i) \qquad X_i \sim \mathcal{U}_\Omega$$

expected value:

discrete

$$E[X] := \sum_{i=1}^{n} x_i \cdot p(x_i)$$

continuous

$$E[X] := \int_\Omega x \cdot p(x)\ dx$$

unbiased estimator is correct on average, for any n

consistent estimator is correct eventually, as $n \to \infty$

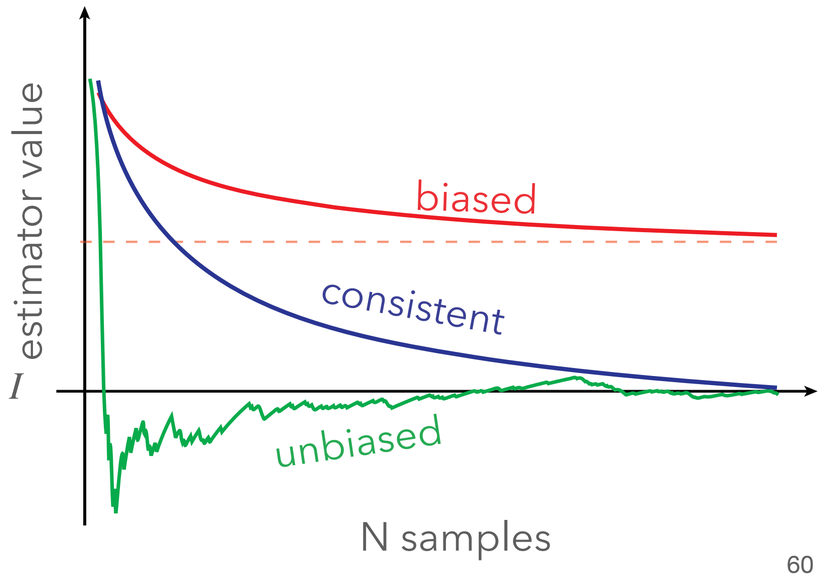
biased estimator converges, but to the incorrect value

unbiased

$$E[\hat{I}_n] = I$$

consistent

$$P\left(\lim_{n\to\infty} E[I_n] = I\right)$$



Basic Monte Carlo estimator is unbiased

Easy to show that the <u>expected value</u> of the basic Monte Carlo estimator with uniformly distributed samples is **unbiased**

integral

$$I := \int_\Omega f(x)\ \mathrm{d}x$$

estimator

$$\hat{I}_n := \frac{|\Omega|}{n} \sum_{i=1}^{n} f(X_i), \quad X_k \sim \mathcal{U}_\Omega$$

$$E[\hat{I}_n] = E\left[\frac{|\Omega|}{n} \sum_{k=1}^{n} f(X_k)\right]$$

definition of $\hat{I}_N$

$$= \frac{|\Omega|}{n} \sum_{k=1}^{n} E\left[f(X_k)\right]$$

linearity of expectation
$E[X+Y] = E[X] + E[Y]$

$$= \frac{|\Omega|}{n} \sum_{k=1}^{n} \int_\Omega f(x)p(x)\ dx$$

definition of expectation

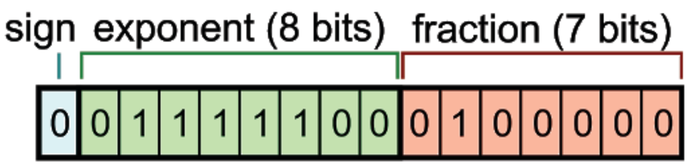
$$= \frac{1}{n} \sum_{k=1}^{n} \int_\Omega f(x)\ dx = I$$

$p(x) = 1/|\Omega|$

Numerical and computational issues

Floating point

- Computers don't operate on real numbers



(16-bit floating point number)

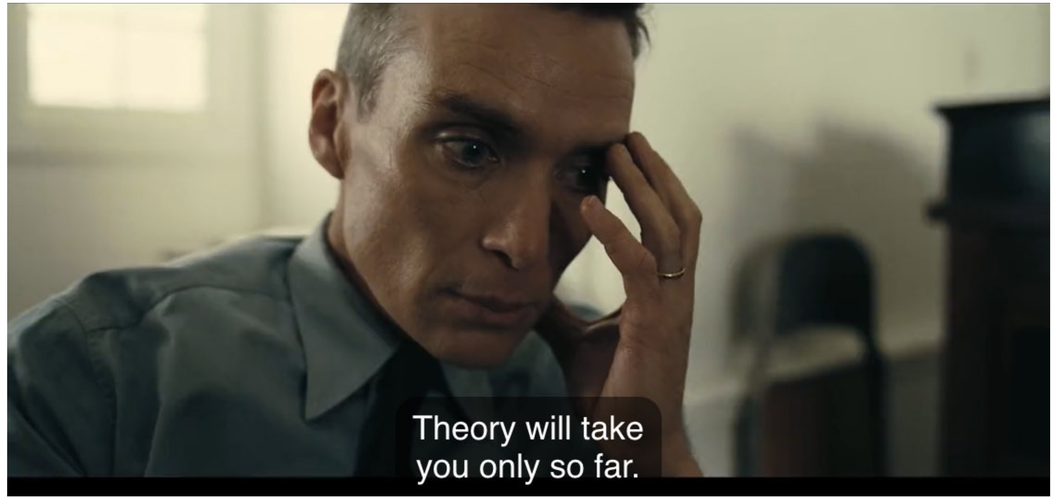
Parallel implementation

- basic Monte Carlo is "embarrassingly parallel"
- Integrand may take a very different amount of work, or use divergent memory accesses

```
float I = 0.0f;
for k=1,..,n {
   x = uniformSample();
   I += f(x)/(float)n;
}
```

Theory and practice is a two way street

- Practical obstacles may motivate changes to mathematical formulation
- E.g., different sampling strategies may be better suited to different architectures



PART 2:
WoS for Laplace Equation

Roadmap for solving Laplace PDE

differential form

Laplace PDE

Find u such that

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

integral representation

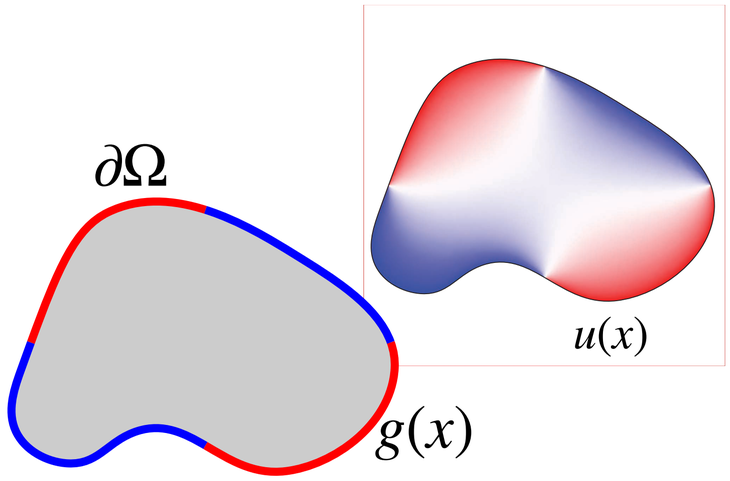
mean value integral

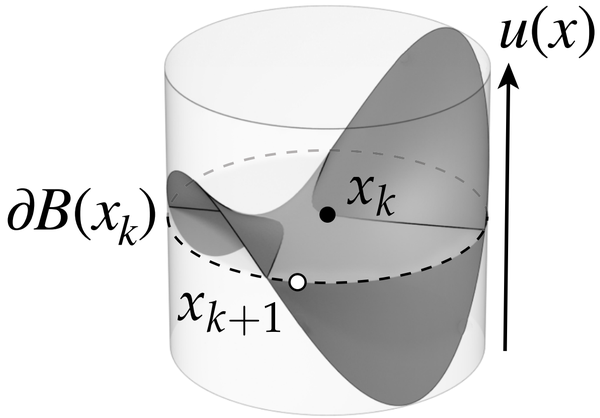
$$u(x_k) = \frac{1}{|\partial B(x_k)|} \int_{\partial B(x_k)} u(y)\, dy$$

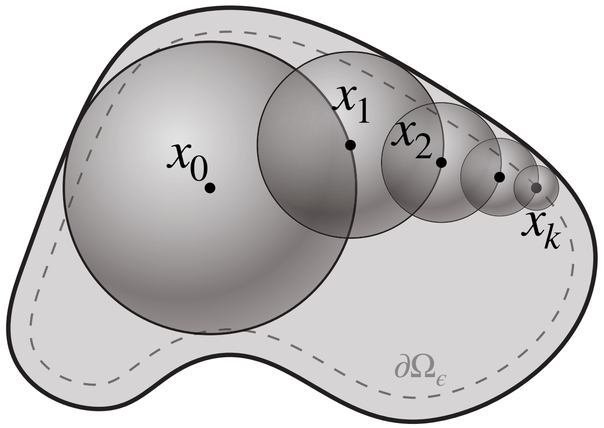
Monte Carlo estimator

Walk on Spheres

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





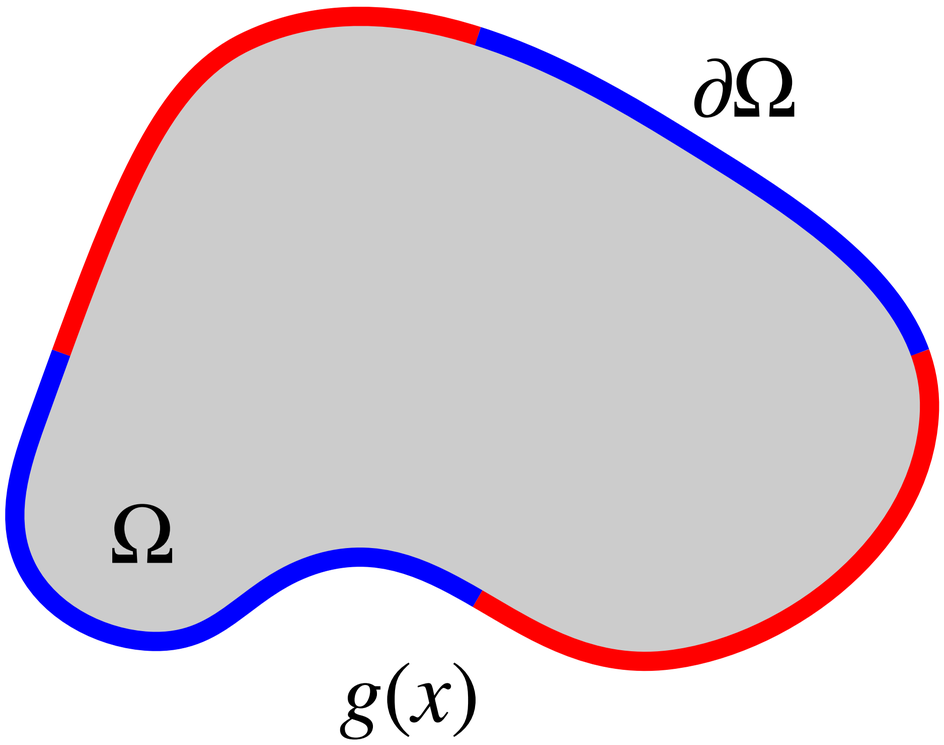


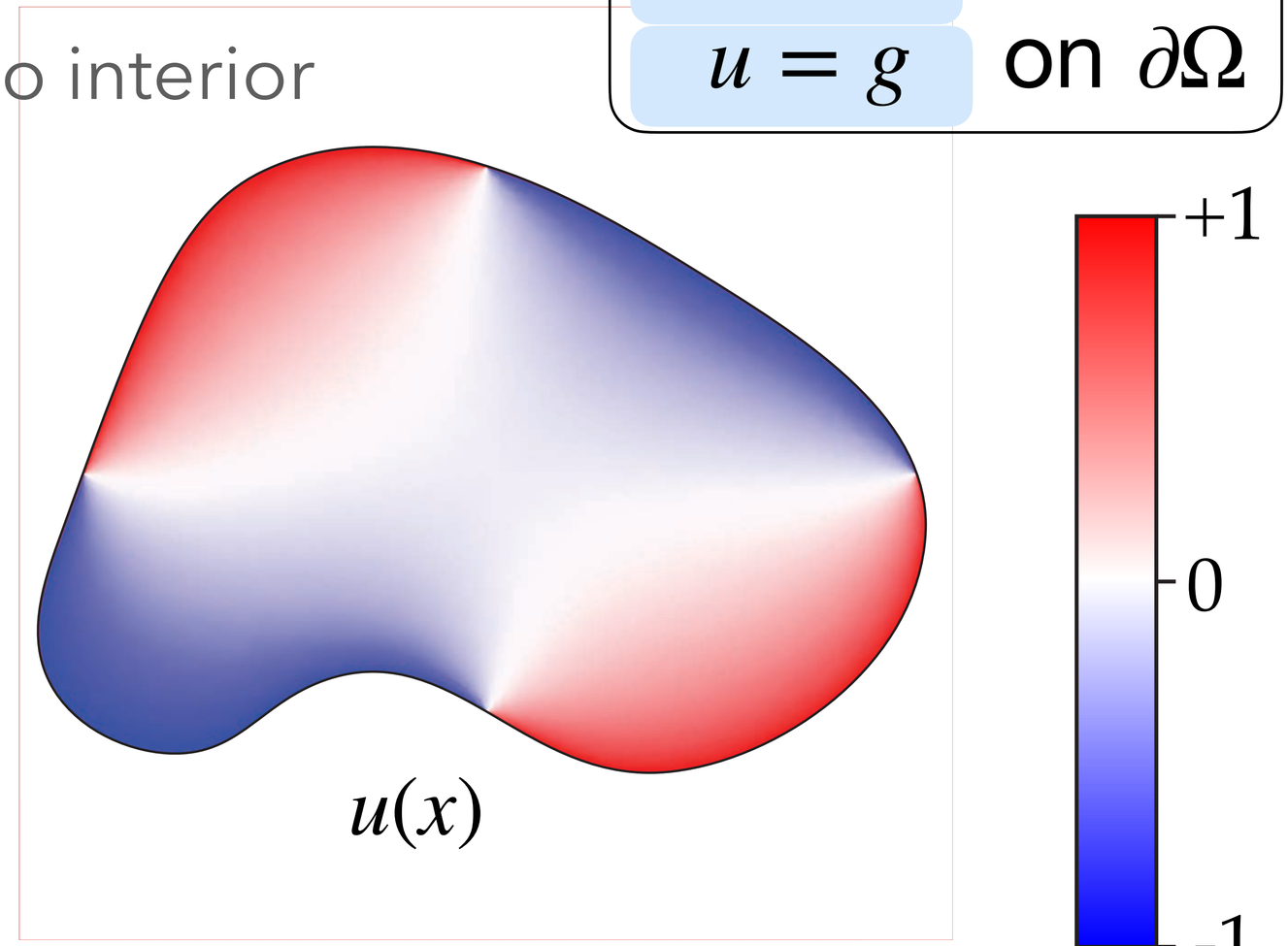
Laplace PDE with Dirichlet boundary condition

given: values on the boundary of a region Ω

find: smooth interpolation into interior

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





Review: Laplacian

$u : \mathbb{R}^n \to \mathbb{R}$ (twice differentiable)

coordinates.

$$\Delta u = \sum_{i=1}^{n} \frac{\partial^2}{\partial x_i^2} u$$

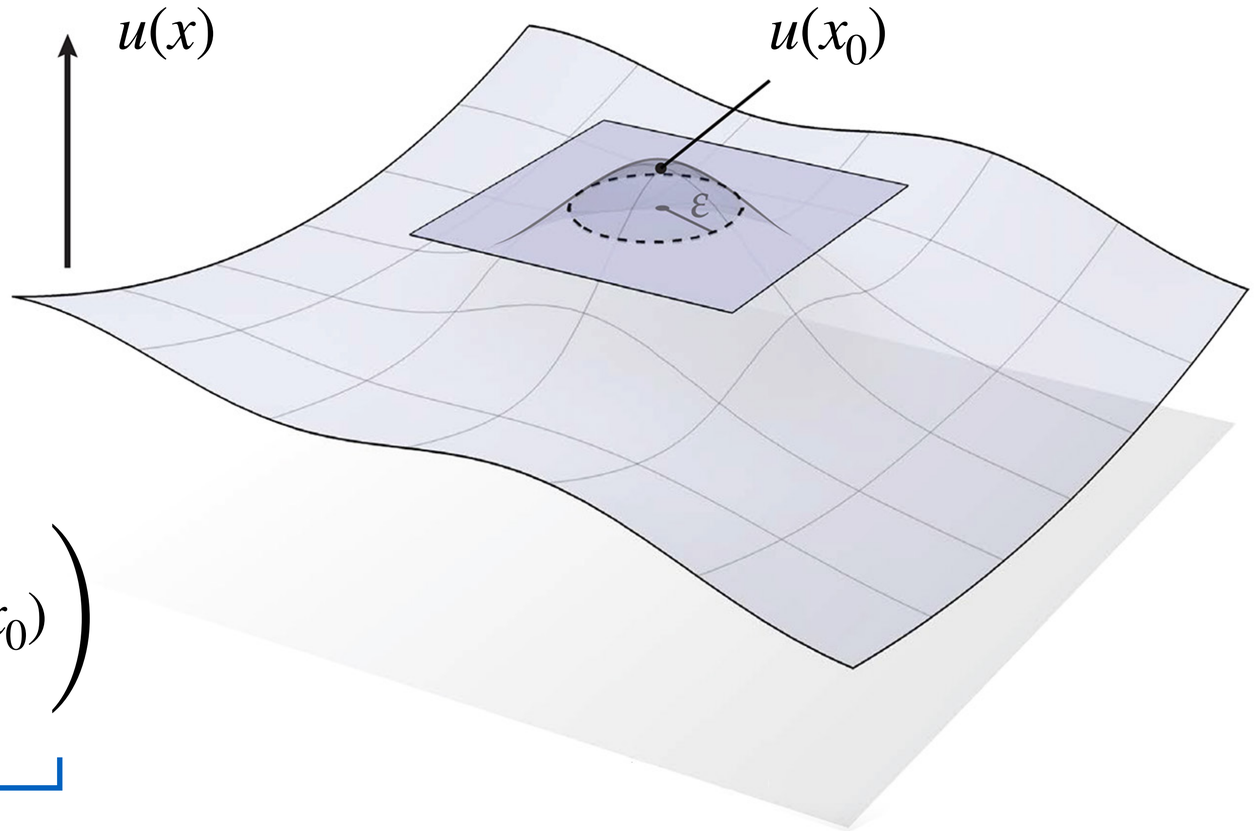
$$\Delta u(x, y) = \frac{\partial^2}{\partial x^2} u(x, y) + \frac{\partial^2}{\partial y^2} u(x, y)$$

differential operators.

$$\Delta u = \nabla \cdot \nabla u = \mathrm{div} \circ \mathrm{grad}\ u$$

Laplacian gives deviation from local average

More intuitively, can think of the Laplacian of a function u as difference between value at a point x_0, and the average value over a small sphere (or ball) around x_0



$$\Delta u(x_0) \propto \lim_{\epsilon \to 0} \frac{1}{\epsilon^2} \left(\underbrace{\frac{1}{|S_\epsilon(x_0)|}}_{\text{sphere area}} \underbrace{\int_{S_\epsilon(x_0)} u(x)\, \mathrm{d}x}_{\text{integral over sphere}} - \underbrace{u(x_0)}_{\text{value at center}} \right)$$

Mean value property of harmonic functions

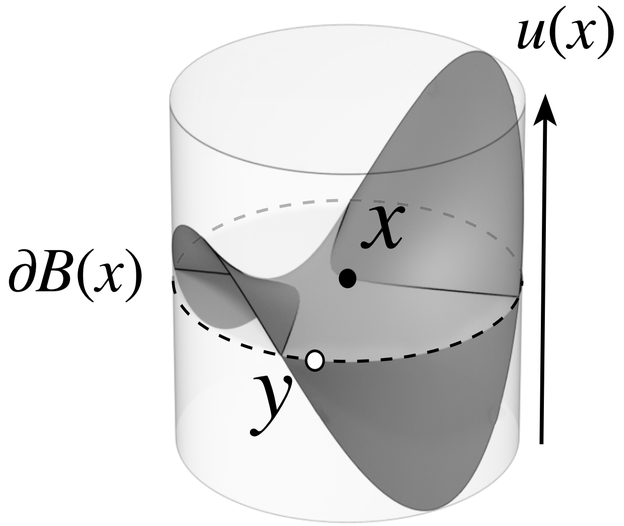
Q: Given this interpretation of the Laplacian, what can we say about the behavior of a ***harmonic function*** (i.e. a function u satisfying $\Delta u = 0$) ?

A: Value at center and average over a sphere are equal.

Not just an approximation, holds <u>exactly</u> for any $B \subset \Omega$ of any radius

mean value property

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



Can we avoid finite-dimensional approximation completely with Monte Carlo?

Monte Carlo estimation of the mean value integral

The mean value property can be evaluated using a basic **Monte Carlo estimator**

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

mean value property

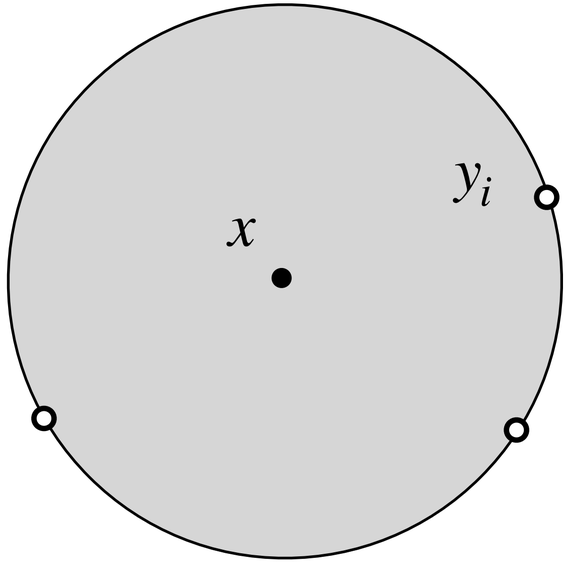
$$u(x) = \frac{1}{|\partial B(x)|} \int_{\partial B(x)} u(y)\, dy$$

mean value estimator

$$\hat{u}(x) = \frac{1}{n} \sum_{i=1}^{n} u(y_i)$$

$u(y_i)$

uniform distribution on sphere $y_i \sim \mathcal{U}_{\partial B(x)}$



Recursive Monte Carlo estimator

In general, we don't know the solution everywhere on a the sphere so we **recursively** evaluate u

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

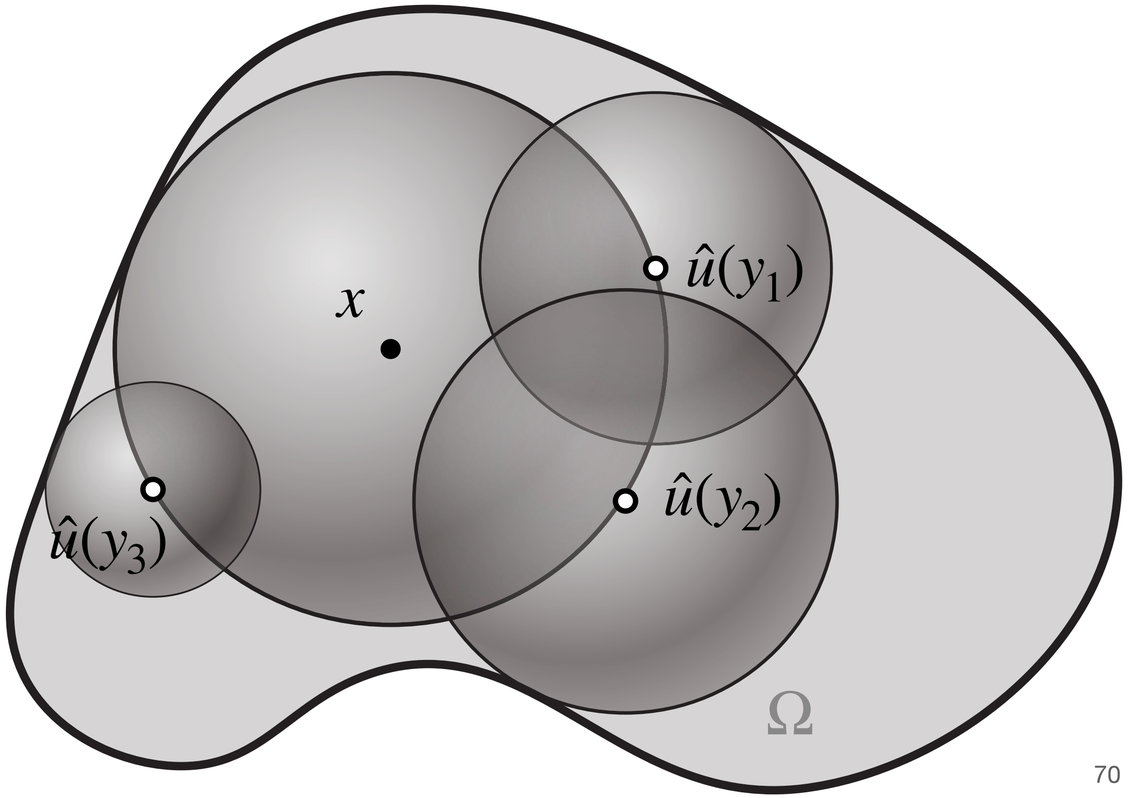
mean value property

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

mean value estimator

$$\hat{u}(x) = \frac{1}{n} \sum_{i=1} \hat{u}(y_i)$$

$$y_i \sim \mathcal{U}_{\partial B(x)}$$



Branching Monte Carlo estimator

Branching is not a feasible strategy since we'll quickly run out of memory trying to store intermediate state

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

mean value property

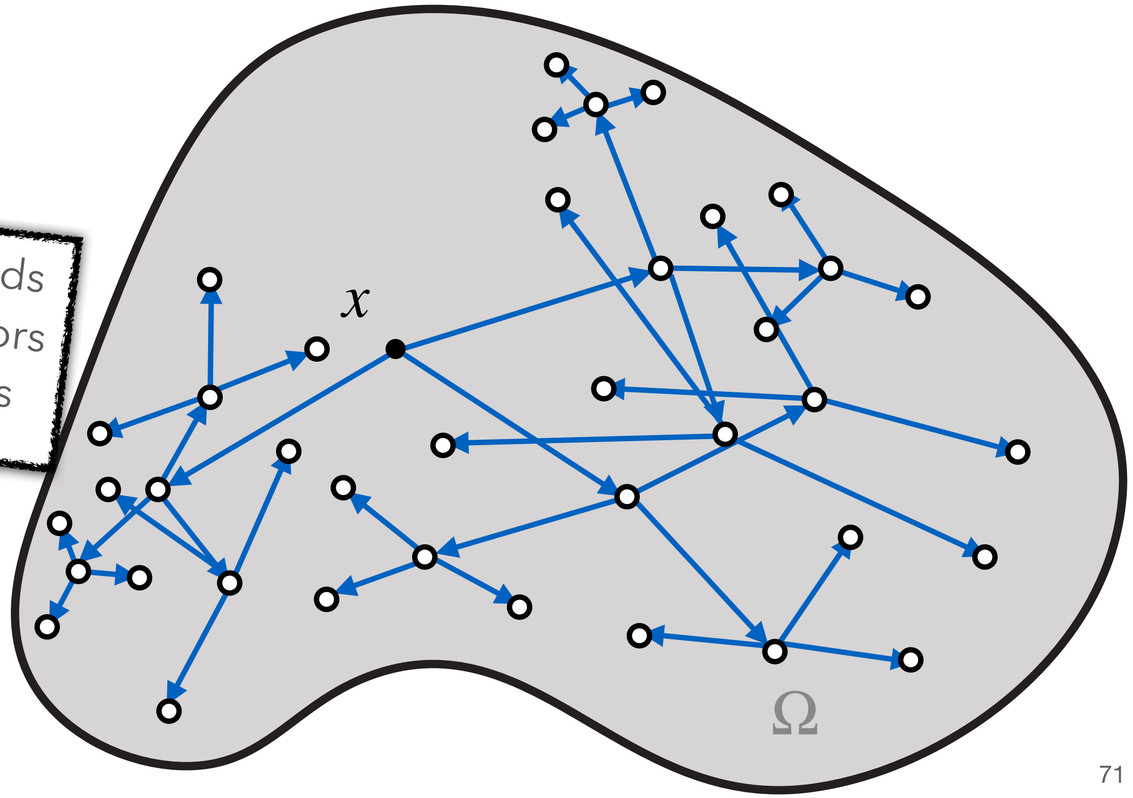
$$u(x) = \frac{1}{|\partial B(x)|} \int_{\partial B(x)} u(y)\, dy$$

mean value estimator

$$\hat{u}(x) = \frac{1}{n} \sum_{i=1} \hat{u}(y_i)$$

$$y_i \sim \mathcal{U}_{\partial B(x)}$$

3 sample estimator leads to **~43 million** estimators after 16 recursive steps



Walk on Spheres

To avoid branching, we only evaluate a <u>single sample</u> at each sphere

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

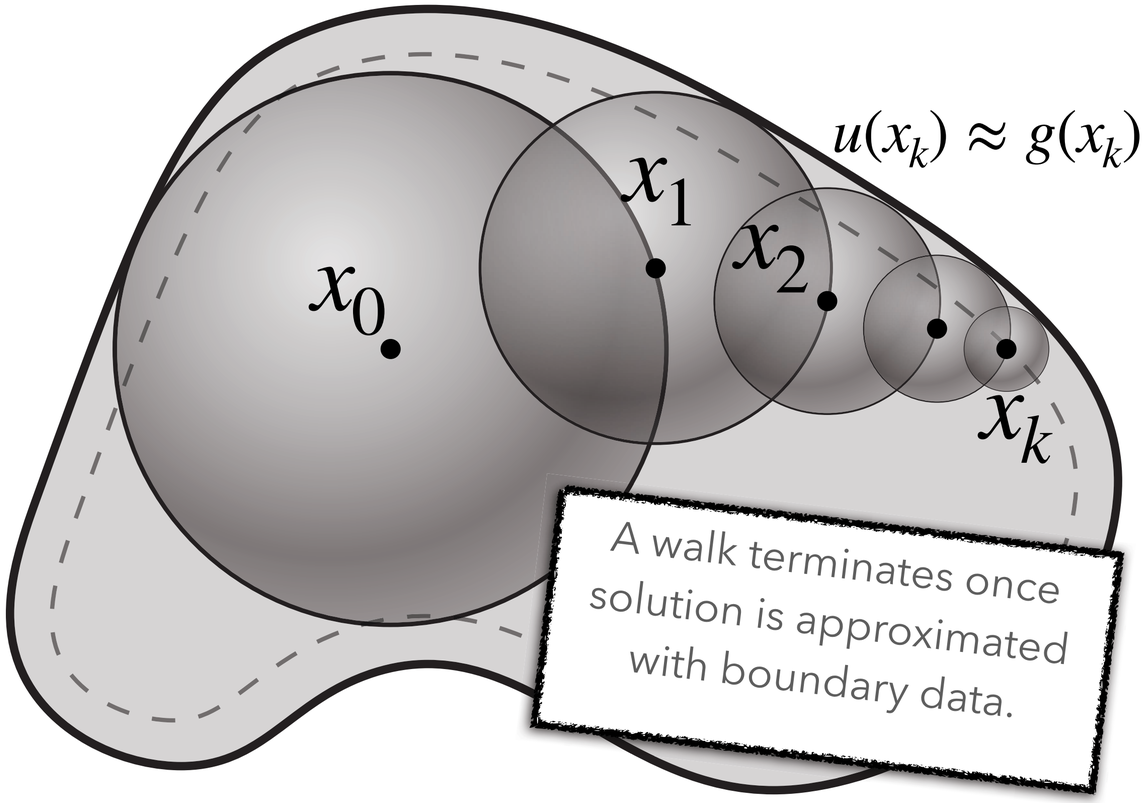
mean value property

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

walk on spheres estimator

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

$$x_i \sim \mathcal{U}_{\partial B(x)}$$



Walk on Spheres algorithm

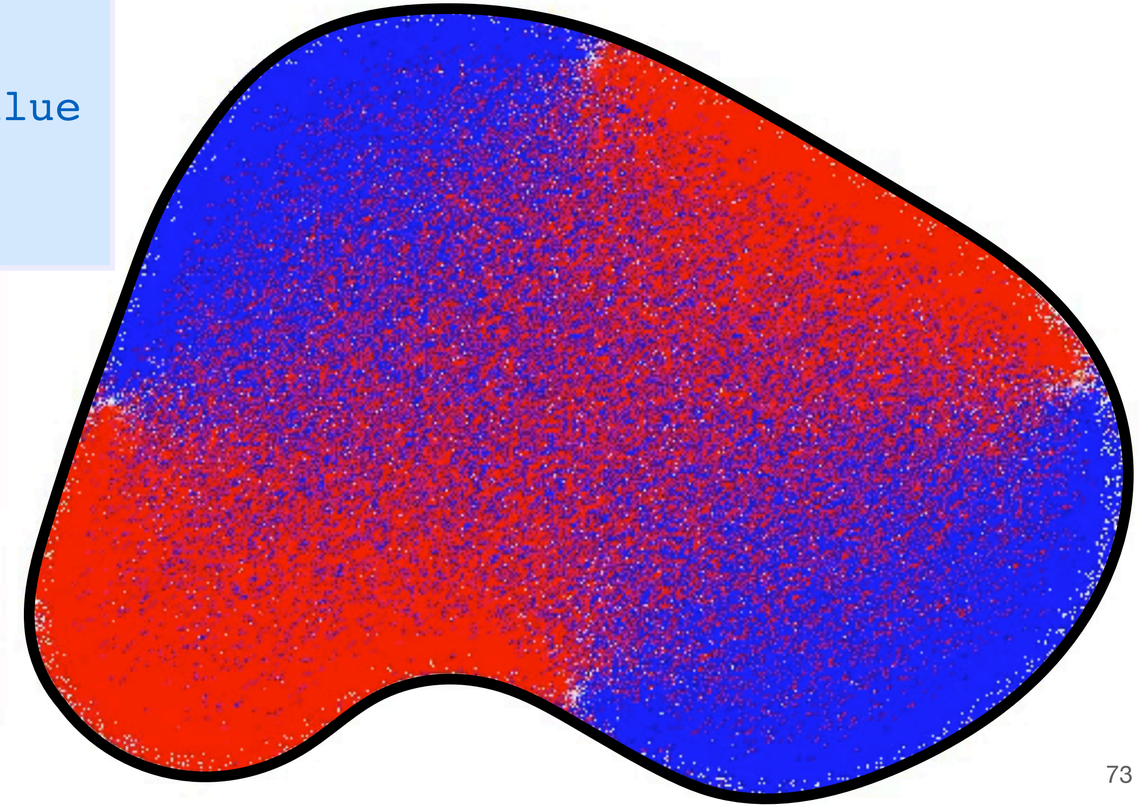
```
u = 0 // solution estimate
for i=1,..,nWalks {
   x = x0 // start a new walk
   do {
      // move to random point on biggest empty sphere
      r = distance(x,∂Ω)
      x = randomSphere(x,r)
   } while(r > ε) // close enough!
   u += g(closestPoint(x,∂Ω)) // sample boundary value
}
return u/nWalks // return average boundary value
```

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

walk on spheres estimator

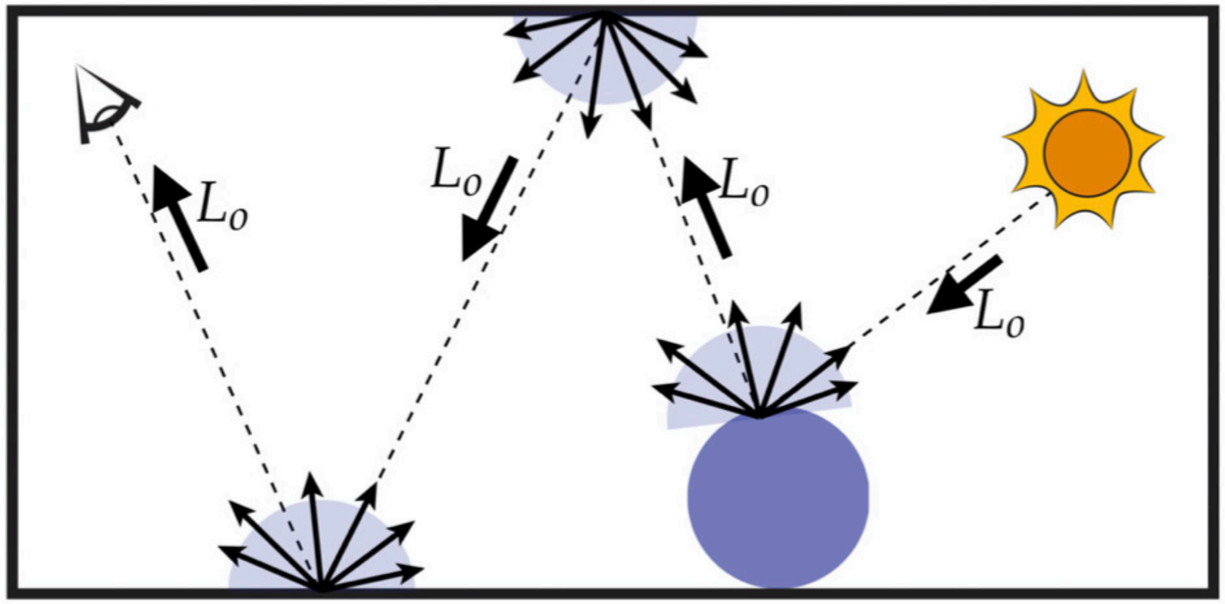
$$\hat{u}(x_i) = \begin{cases} \hat{u}(x_{i+1}) & \text{if } x_{i+1} \notin \partial\Omega_\epsilon \\ g(x_{i+1}) & \text{otherwise} \end{cases}$$

$$x_i \sim \mathcal{U}_{\partial B(x)}$$

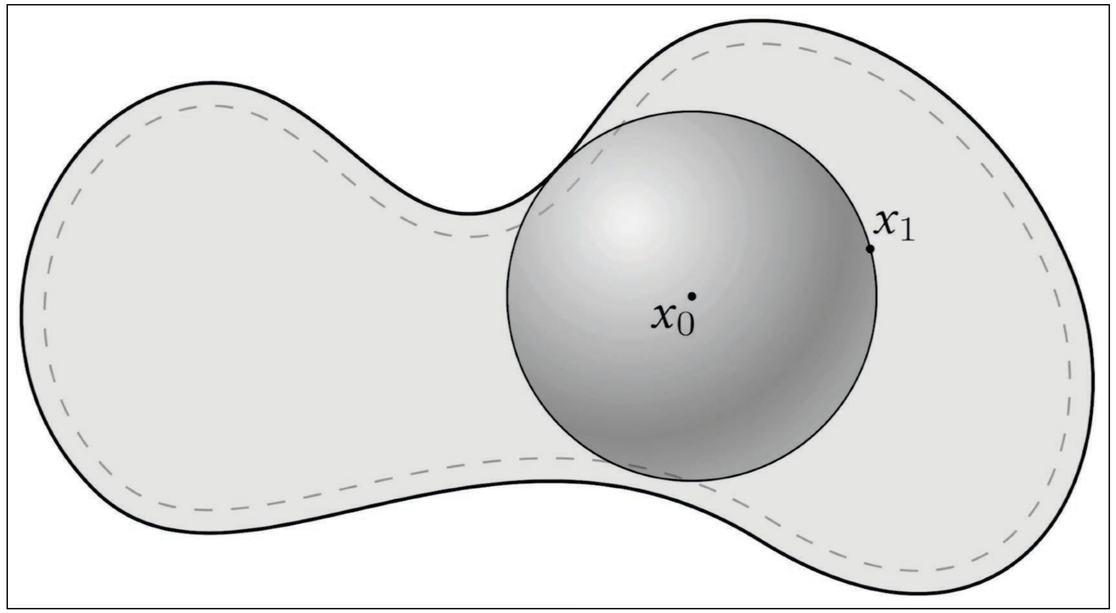


Recursive estimators

Recursive random walks for solving the Laplace equation $\Delta u = 0$



ray tracing



walk on spheres [Muller 1956]

Warning about convergence

Earlier we showed that Monte Carlo estimators of integrals converge

$$I = \int_{\Omega} f(x)\ \mathrm{d}x \qquad \hat{I}_n = \frac{|\Omega|}{n} \sum_{i=1}^{N} f(X_i)$$

Q: Are Monte Carlo estimates of arbitrarily nested integrals well defined?

$$u(x_k) = \frac{1}{|\partial B_k|} \int_{\partial B_k} u(x_{k+1})\ dx_{k+1}$$

A: Need to make sure that the integral <u>exists</u>…

$$u(x) = \lim_{k \to \infty} \left[\prod_{i=0}^{k} \frac{1}{|\partial B_i|} \right] \int_{\partial B_0} \int_{\partial B_1} .. \int_{\partial B_k} u(x_{i+1})\ dx_k \ldots dx_2 dx_1$$

Convergence of WoS – integral viewpoint

Q: For a matrix $A \in \mathbb{R}^{n \times n}$ how do we check whether $\lim_{k \to \infty} A^k x$ exists?

A: Just check that A cannot make any vector "bigger" at each step

matrix convergence

$$\|Ax\| < \|x\| \quad \forall x \in \mathbb{R}^n$$

Same approach for mean value integral, which uses an <u>operator norm</u>

linear operator convergence

$$\|Lu(x)\|_{op} < \|u(x)\|_{op} \;\; \forall u \in \mathbb{R}^n \to \mathbb{R}$$

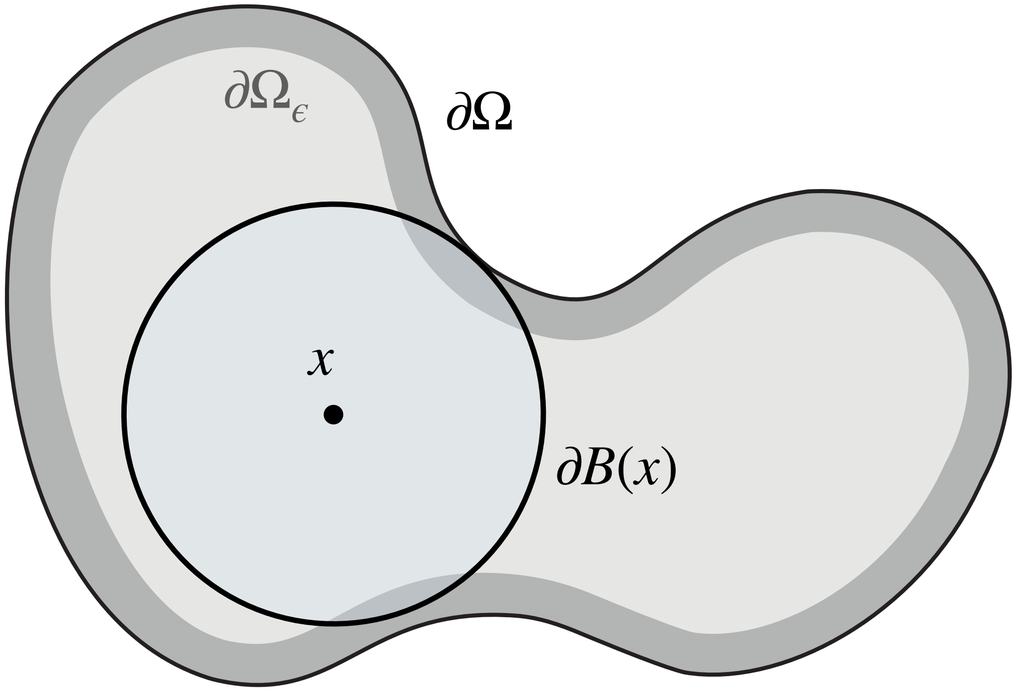
integral operator norm

$$\|u\|_{op} := \int_{\partial\Omega} |\, u(x) \,|\; dx$$

Non-convergence for mean value operator

Mean value integral is an "averaging operator" applied to the solution $u(x)$ on the boundary of the sphere

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



convergence criteria

$$\|Lu(x)\|_{op} < \|u(x)\|_{op} \iff \|L\|_{\text{op}} < 1$$

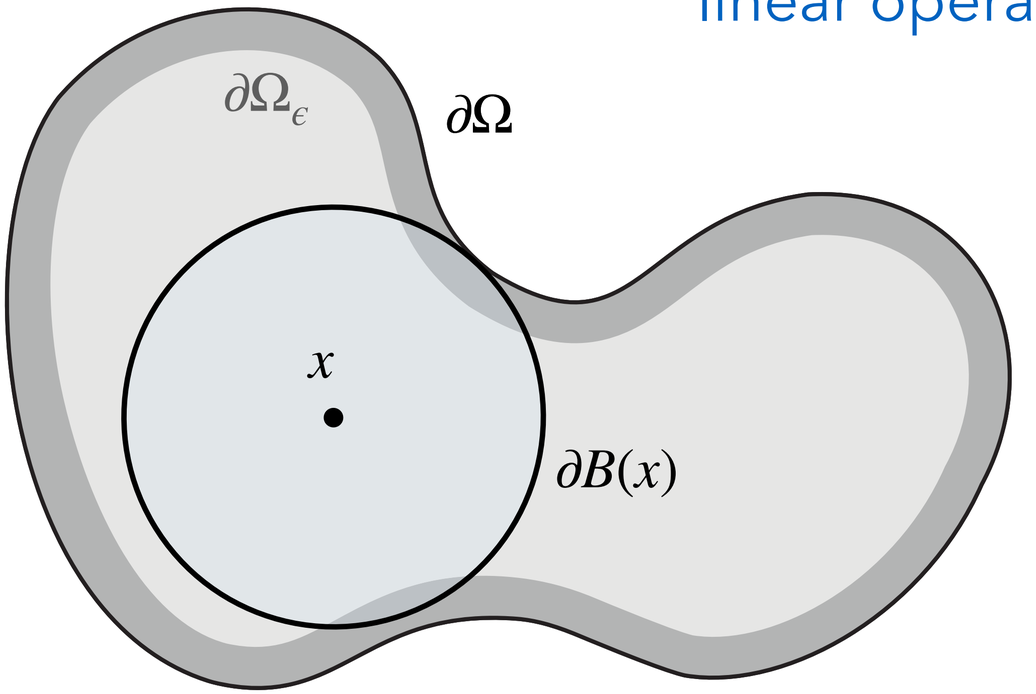
$$\|L\|_{\text{op}} = \frac{1}{|\partial B(x)|} \int_{\partial B(x)} \mathrm{d}x = 1$$

ϵ-shell to the rescue

Integral operator L for walk on spheres isn't quite the "averaging operator"– also accounts for the contribution from Dirichlet boundary from ϵ**-shell**

$$Lu(x) := \frac{1}{|\partial B(x)|}\int_{\partial B(x)\setminus\partial\Omega_\epsilon} u(y)\,dy + \frac{1}{\partial B(x)}\int_{\partial B(x)\cap\partial\Omega_\epsilon} g(y)\,dy$$

linear operator $L_\epsilon u$ constant term b



convergence criteria

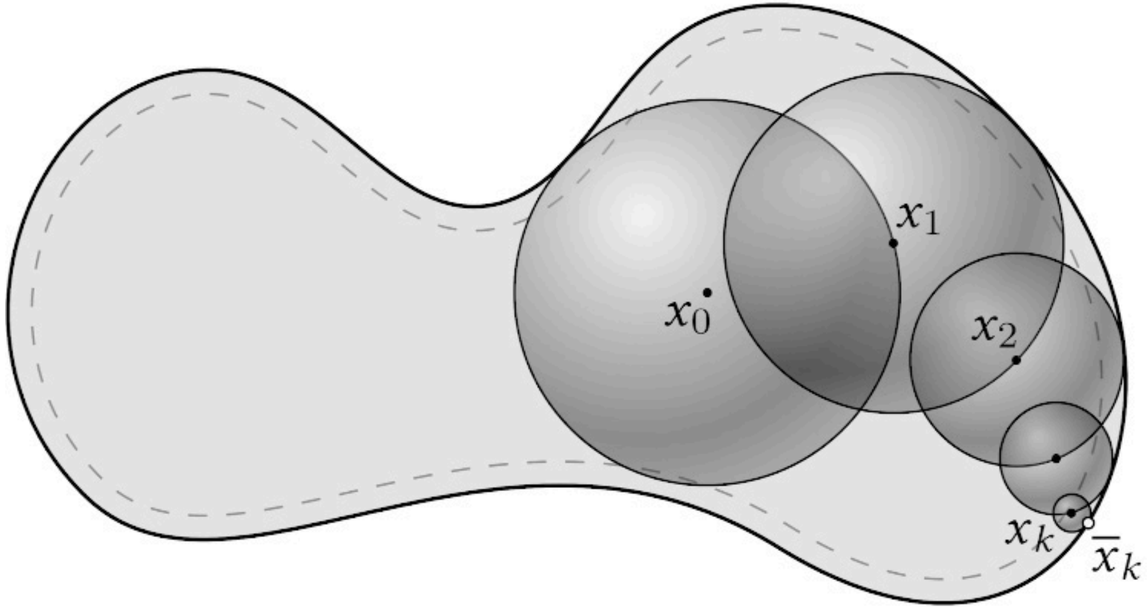
$$\|Lu(x)\|_{op} < \|u(x)\|_{op} \iff \|L\|_{\text{op}} < 1$$

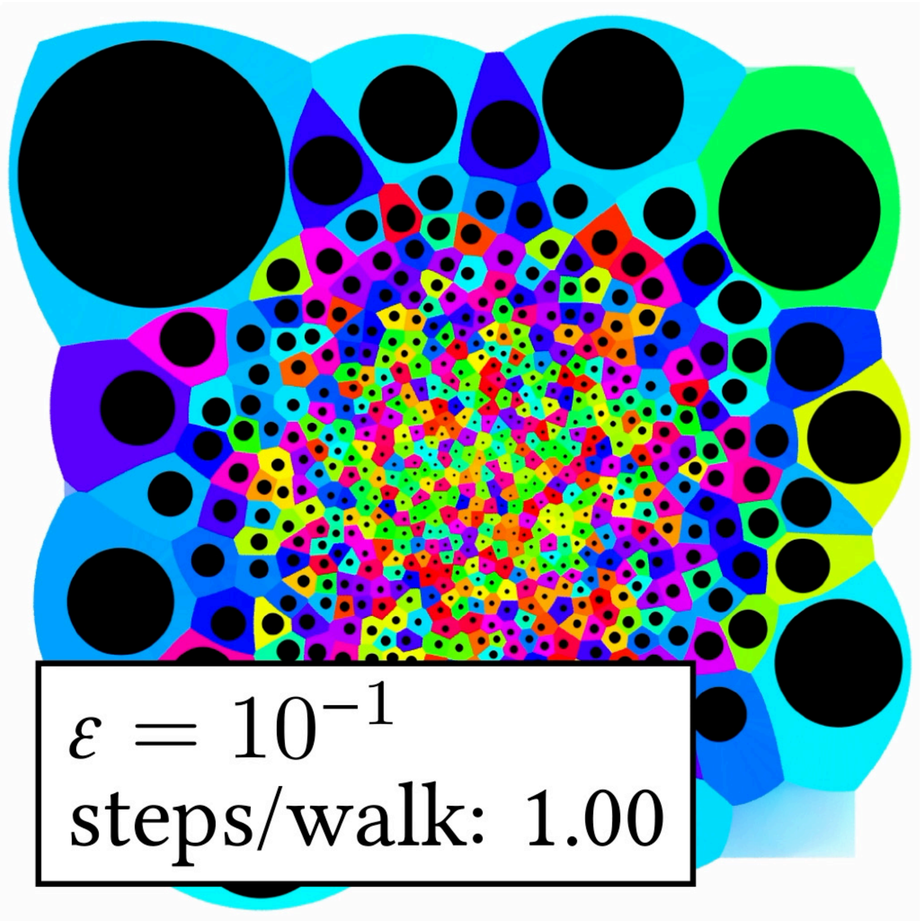
$$\|L\|_{\text{op}} = \frac{1}{|\partial B(x)|}\int_{\partial B(x)} \mathrm{d}x = 1$$

$$\|L_\epsilon\|_{\text{op}} = \frac{1}{|\partial B(x)|}\int_{\partial B(x)\setminus\Omega_\epsilon} \mathrm{d}x < 1$$

Stopping tolerance ε for Dirichlet boundary

Introduces minimal bias and has little impact on performance





Efficiency of Walk on Spheres

Theorem 2. *Let Ω be a bounded α-thick domain in $\mathbb{R}^d$. Then the expected rate of convergence of the WoS from any $x \in \Omega$ until termination at distance $< \varepsilon$ to the boundary is given by the following table:*

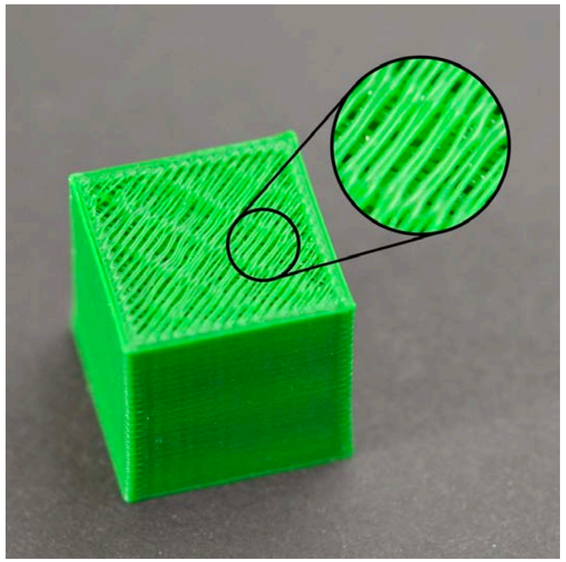
Theorem: If the domain boundary $\partial\Omega$ is smooth, or the domain Ω is convex, then WoS reaches the boundary in $O(\log 1/\epsilon)$ steps, on average.

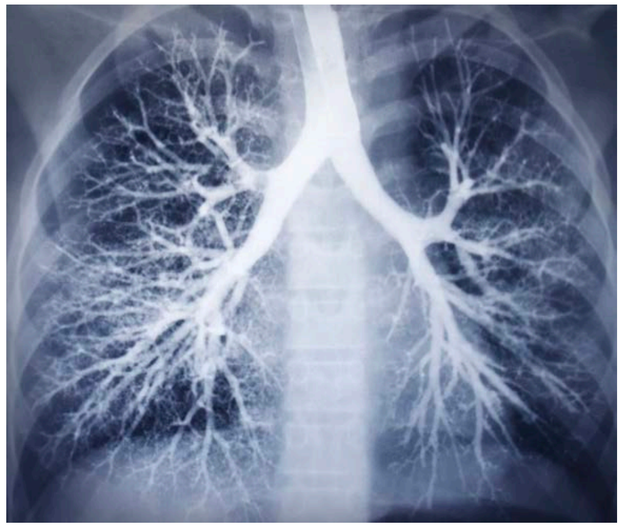
Moreover, the rates of convergence above are tight. Formally, let $f_\alpha(1/\varepsilon)$ be the growth rate asymptotically given by the formulas in (3). For any α and for any $g(1/\varepsilon)$ such that $g(1/\varepsilon) = o(f_\alpha(1/\varepsilon))$, there is an α-thick domain Ω_g^α with some thickness C such that the rate of convergence of the WoS algorithm on Ω_g^α is asymptotically slower than $g(1/\varepsilon)$ for a sequence $\{\varepsilon_n\}_{n=1}^\infty$ of ε's that converge to 0.

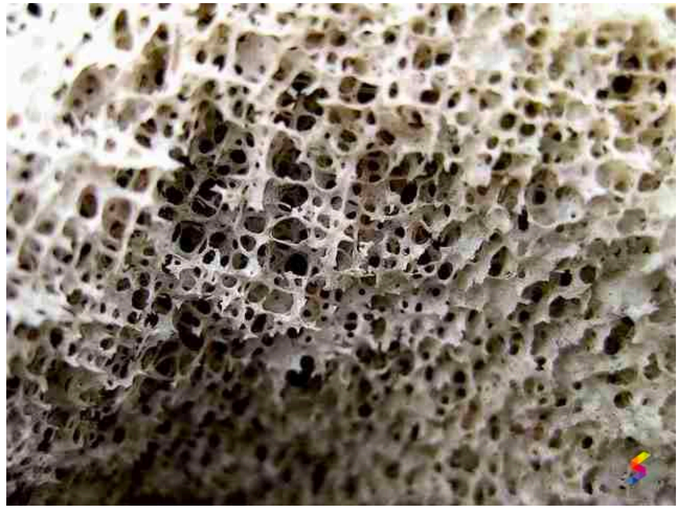
Binder & Braverman, "The Rate of Convergence of the Walk on Spheres Algorithm" (2012)

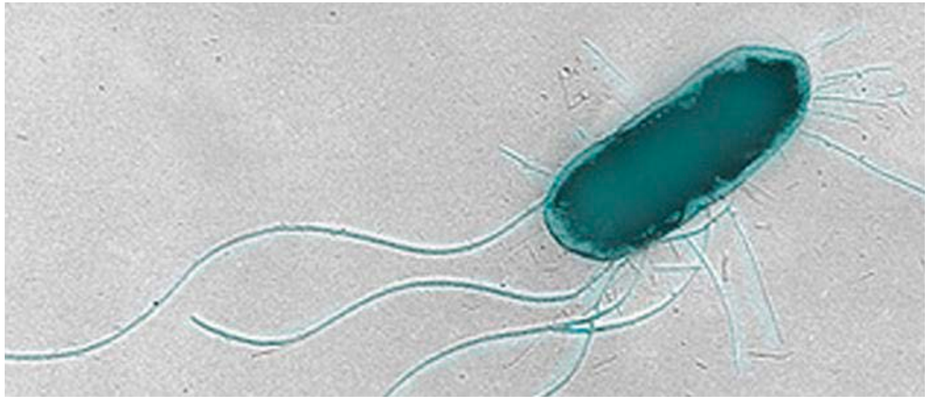
Stochastic vs. Deterministic Methods

How much does it cost to capture fine-scale features, of size $O(\epsilon)$



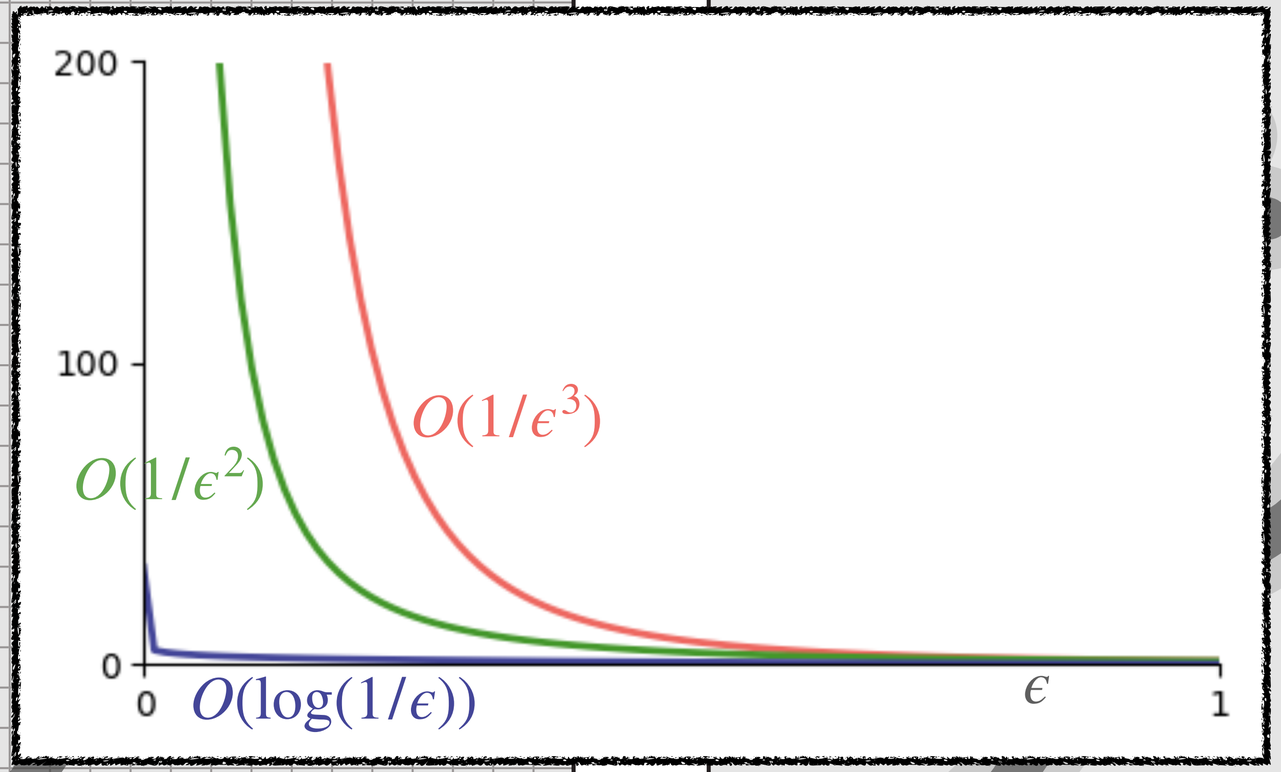






Stochastic vs. Deterministic Methods

$O(1/\varepsilon^d)$ grid cells



$O(\log 1/\varepsilon)$ steps

finite differences

walk on spheres

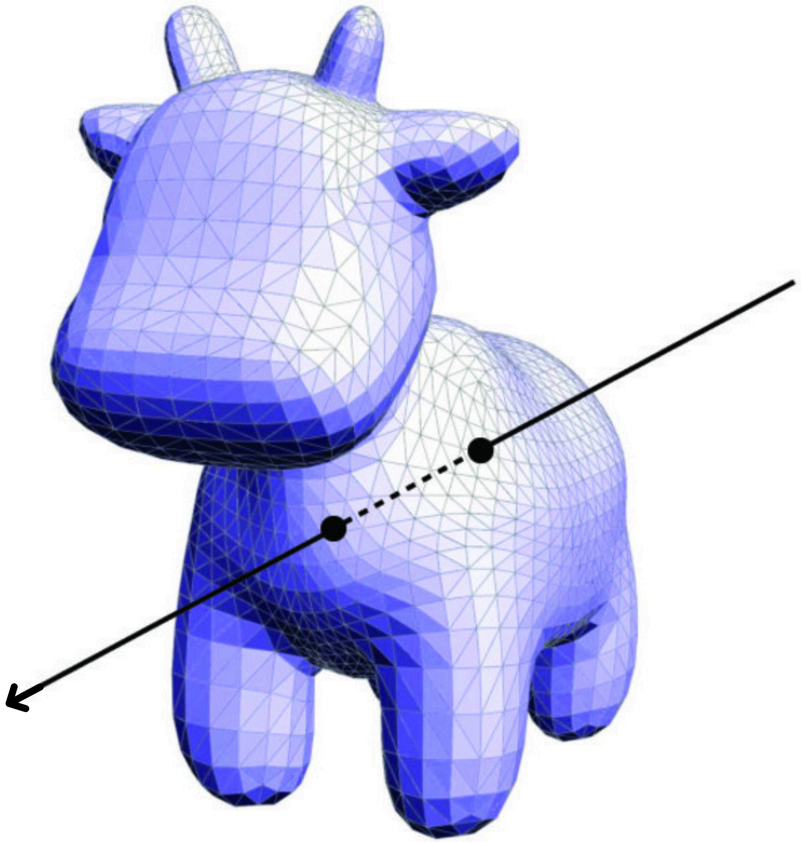
Closest Point Queries

Q: How expensive is a single step of walk on spheres?

A: Depends on the cost of computing the distance to the closest point

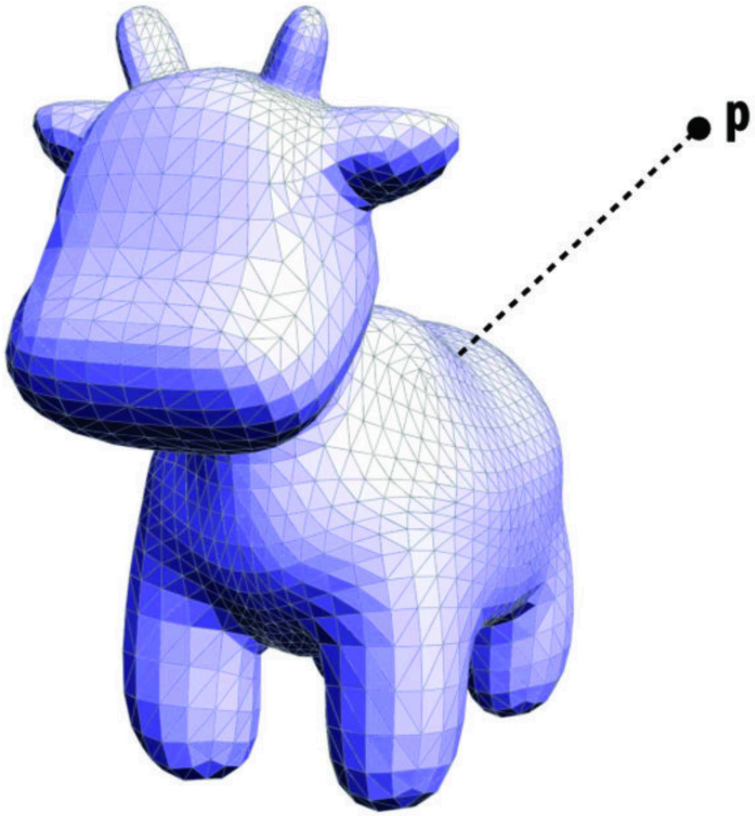
Geometric queries in Monte Carlo methods

rendering



Ray Intersection Query

walk on spheres



Distance Query

Closest point queries

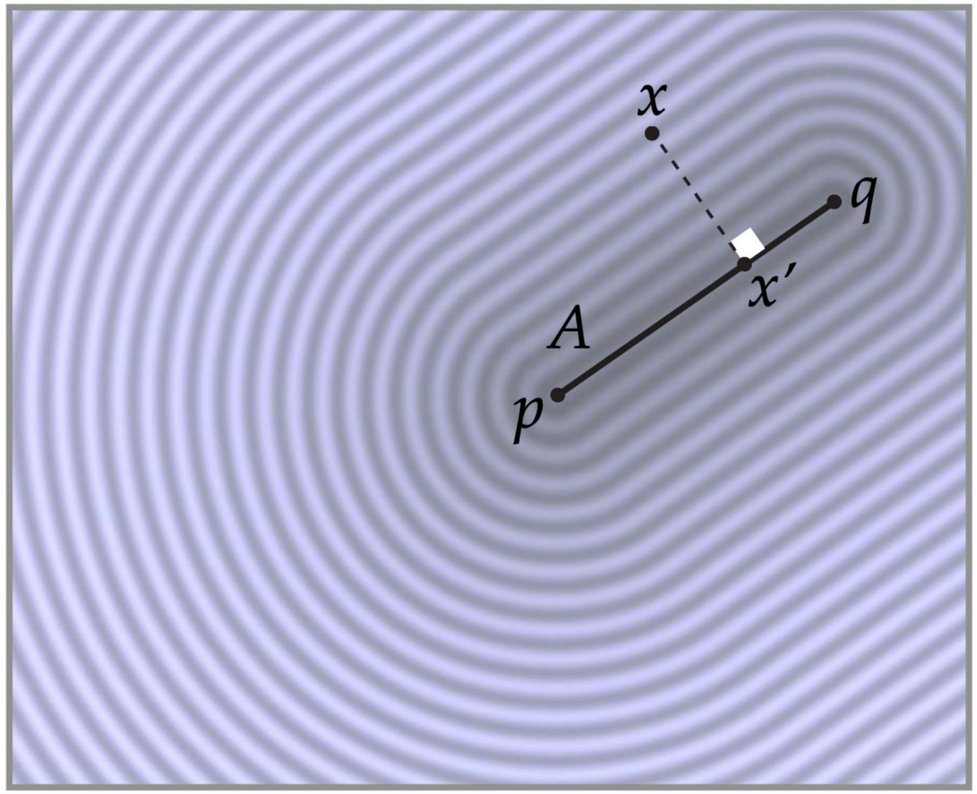
For query point x, find closest point x' on domain boundary $\partial\Omega$

Example. Line segment ($\partial\Omega = A$)

$$t := (x - p) \cdot (q - p) / |p - q|^2$$

$$x' = \begin{cases} p, & t < 0 \\ q, & t > 0, \\ (1 - t)p + tq, & \text{otherwise} \end{cases}$$

$$d(x, A) = |x - x'|$$



Closest point queries

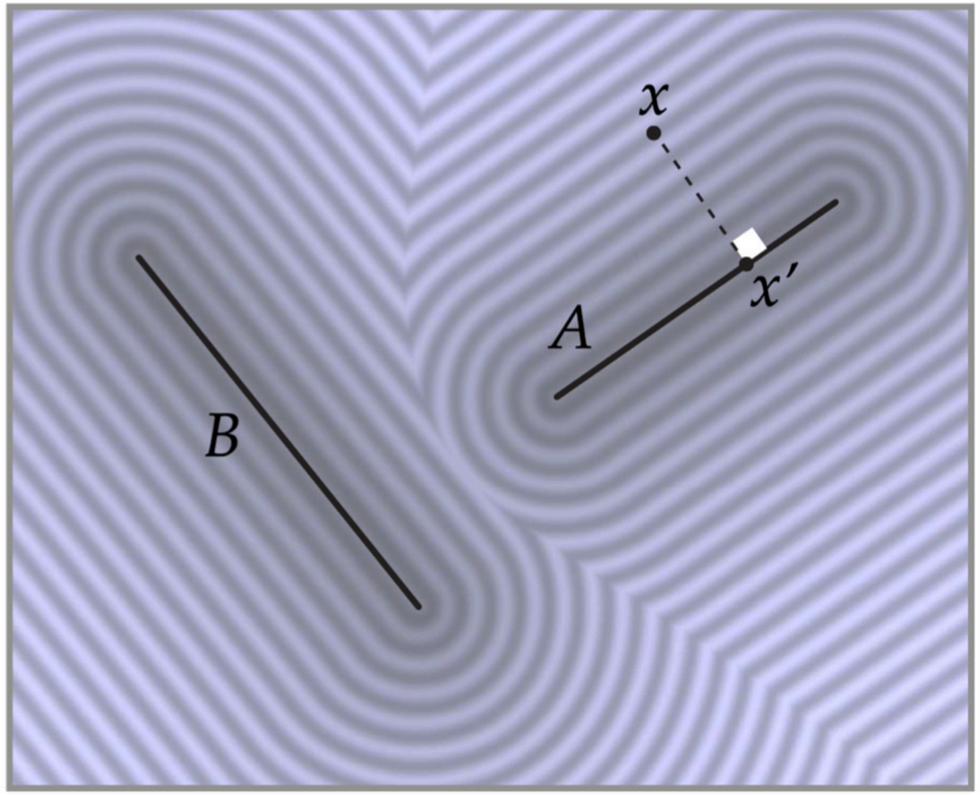
For query point x, find closest point x' on domain boundary $\partial\Omega$

Example. Line segment ($\partial\Omega = A$)

$$d(x, A) = |x - x'|$$

Example. Two line segment ($\partial\Omega = A \cup B$)

$$d(x, A \cup B) = \min(d(x, A), d(x, B))$$



Closest point queries

For query point x, find closest point x' on domain boundary $\partial\Omega$

Example. Line segment ($\partial\Omega = A$)

$$d(x, A) = |x - x'|$$

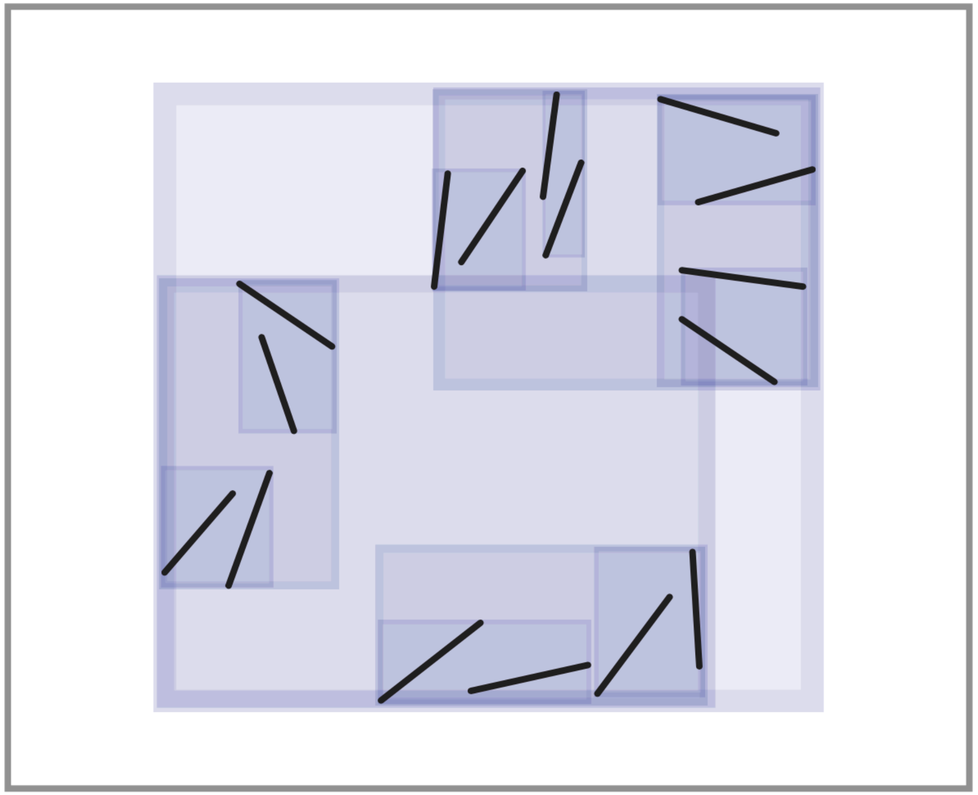
Example. Two line segment ($\partial\Omega = A \cup B$)

$$d(x, A \cup B) = \min(d(x, A), d(x, B))$$

Example. Large number of line segments

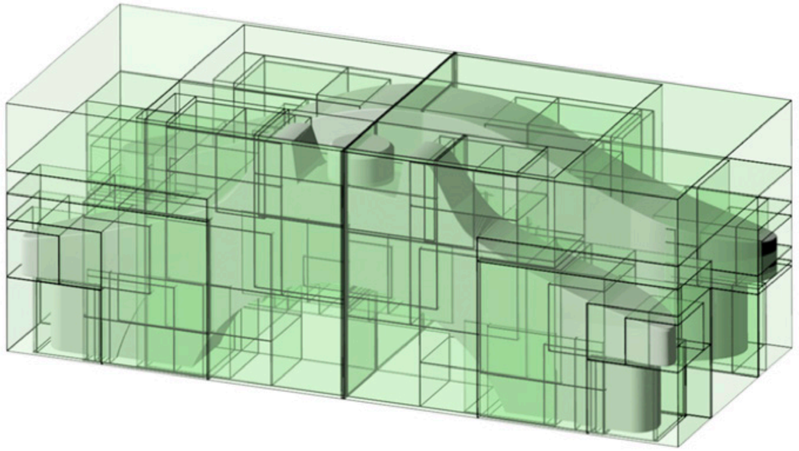
Build *bounding volume hierarchy* (BVH)

amortized cost of query is $O(\log n)$

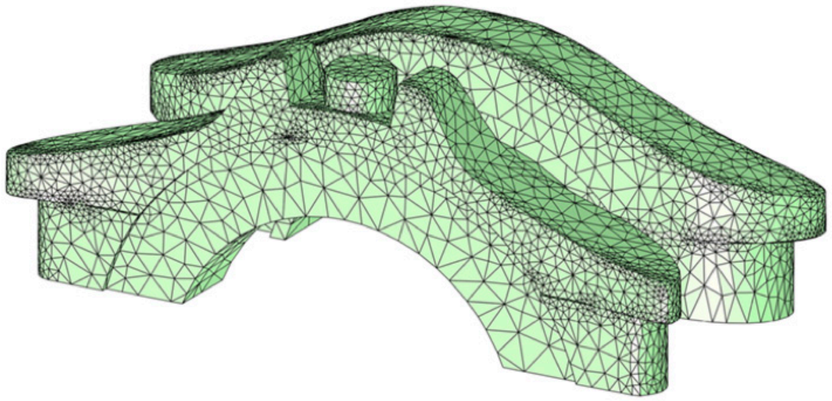


Benefits of Monte Carlo – Lightweight data-structure

input
boundary mesh



build BVH for WoS
few milliseconds



FEM mesh (FastTetWild)
1 hour 25 minutes

Geometric generality

Walk on spheres works any geometry that supports empty sphere queries, for example:

explicit **implicit**

mesh **signed distance** **neural** **harmonic**

Spelunking the Deep: Guaranteed Queries on General Neural Implicit Surfaces via Range Analysis

NICHOLAS SHARP, University of Toronto, Canada
ALEC JACOBSON, University of Toronto, Adobe Research, Canada

Neural implicit representations, which encode a surface as the level set of a neural network applied to spatial coordinates, have proven to be remarkably effective for optimizing, compressing, and generating 3D geometry. Although these representations are easy to fit, it is not clear how to best evaluate geometric queries on the shape, such as intersecting against a ray or finding a closest point. The predominant approach is to encourage the network to have a signed distance property. However, this property typically holds only approximately, leading to robustness issues, and holds only at the conclusion of training, inhibiting the use of queries in loss functions. Instead, this work presents a new approach to perform queries directly on *general* neural implicit functions for a wide range of existing architectures. Our key tool is the application of range analysis to neural networks, using automatic arithmetic rules to bound the output of a network over a region; we conduct a study of range analysis on neural networks, and identify variants of affine arithmetic which are highly effective. We use the resulting bounds to develop geometric queries including ray casting, intersection testing, constructing spatial hierarchies, fast mesh extraction, closest-point evaluation, evaluating bulk properties, and more. Our queries can be efficiently evaluated on GPUs, and offer concrete accuracy guarantees even on randomly-initialized networks, enabling their use in training objectives and beyond. We also show a preliminary application to inverse rendering.

CCS Concepts: • **Computing methodologies** → **Shape analysis**; **Shape representations**; • **Mathematics of computing** → *Interval arithmetic.*

Additional Key Words and Phrases: implicit surfaces, neural networks, range analysis, geometry processing

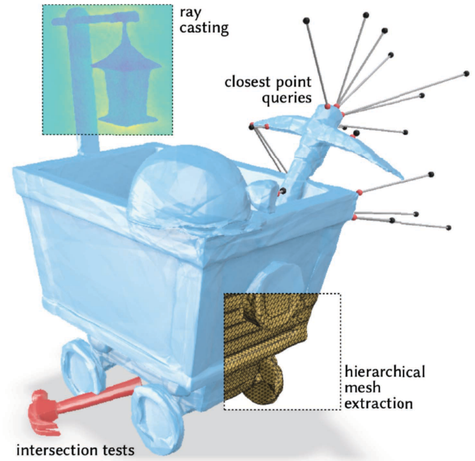


Fig. 1. Our method enables geometric queries on neural implicit surfaces, without relying on fitting a signed distance function. Several queries are shown here on a neural implicit occupancy function encoding a mine cart. These operations open up new explorations of deep implicit surfaces.

Ray Tracing Harmonic Functions

MARK GILLESPIE, Carnegie Mellon University, USA
DENISE YANG, Carnegie Mellon University, USA and Pixar Animation Studios, USA
MARIO BOTSCH, TU Dortmund University, Germany
KEENAN CRANE, Carnegie Mellon University, USA

Sphere tracing is a fast and high-quality method for visualizing surfaces encoded by signed distance functions (SDFs). We introduce a similar method for a completely different class of surfaces encoded by *harmonic functions*, opening up rich new possibilities for visual computing. Our starting point is similar in spirit to sphere tracing: using conservative *Harnack bounds* on the growth of harmonic functions, we develop a *Harnack tracing* algorithm for visualizing level sets of harmonic functions, including those that are angle-valued and exhibit singularities. The method takes much larger steps than naïve ray marching, avoids numerical issues common to generic root finding methods and, like sphere tracing, needs only perform pointwise evaluation of the function at each step. For many use cases, the method is fast enough to run real time in a shader program. We use it to visualize smooth surfaces directly from point clouds (via Poisson surface reconstruction) or polygon soup (via generalized winding numbers) without linear solves or mesh extraction. We also use it to visualize nonplanar polygons (possibly with holes), surfaces from architectural geometry, mesh "exoskeletons", and key mathematical objects including knots, links, spherical harmonics, and Riemann surfaces. Finally we show that, at least in theory, Harnack tracing provides an alternative mechanism for visualizing arbitrary implicit surfaces.

CCS Concepts: • **Computing methodologies** → **Ray tracing**; *Shape analysis*; • **Mathematics of computing** → *Numerical analysis.*

Additional Key Words and Phrases: Ray tracing, sphere tracing, implicit surfaces, harmonic function, Harnack inequality

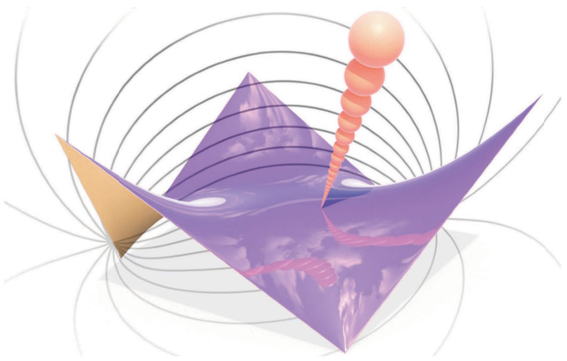
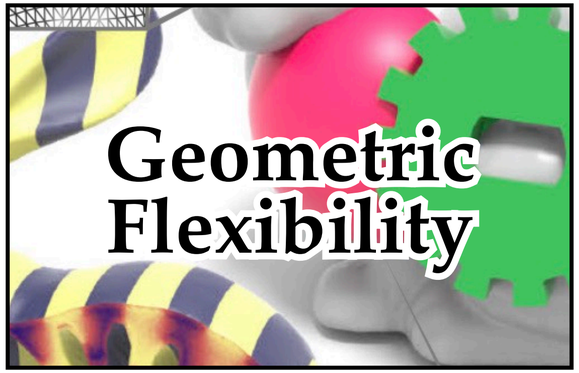
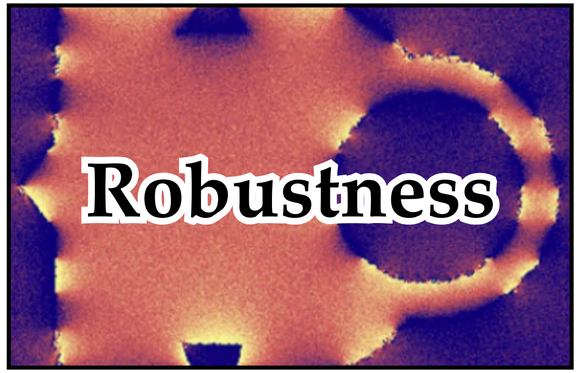


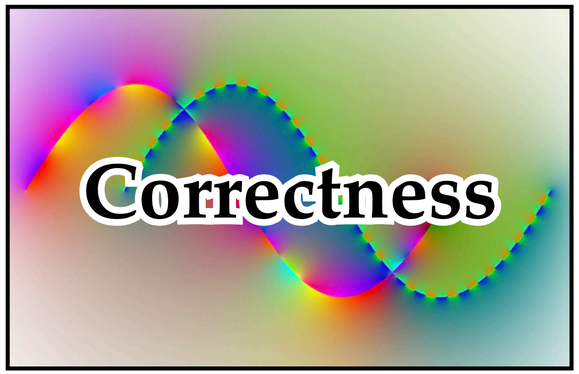
Fig. 1. We introduce a ray tracing algorithm for a novel class of surfaces defined by level sets of harmonic functions. Here for instance we directly visualize a nonplanar polygon which has no well-defined inside or outside—and hence cannot be represented by an ordinary implicit function or SDF. Isolines depict a 2D slice of the harmonic function; spheres show conservative *Harnack bounds* along a ray. Note the smooth reflection lines, even near edges and vertices where the function is highly singular.

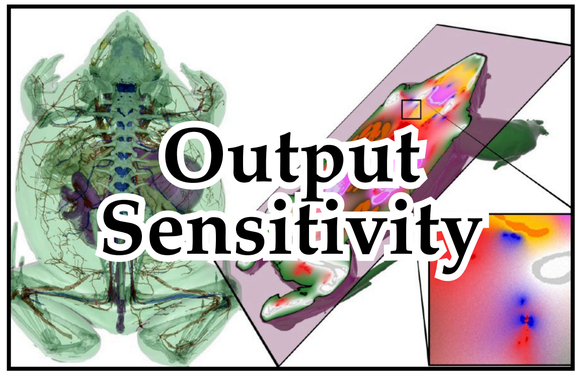
Benefits of Monte Carlo Methods

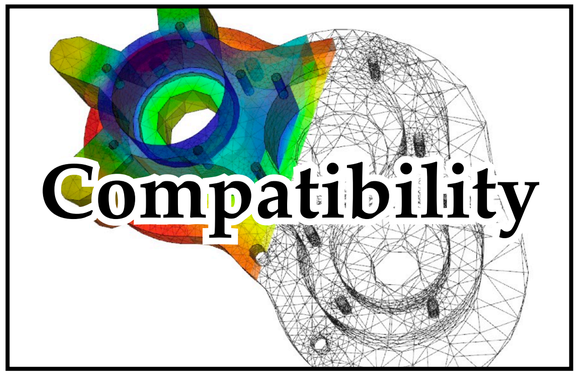
Monte Carlo PDE solvers provides many of the same benefits:









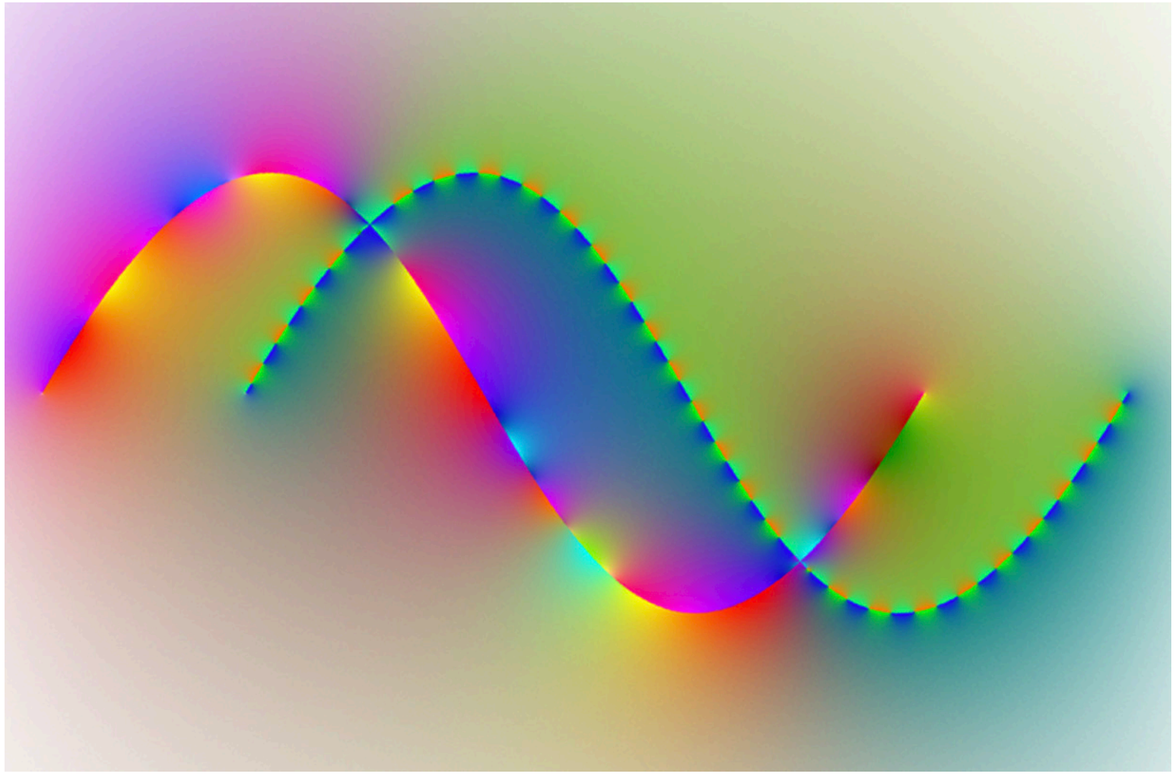


Benefits of Monte Carlo – Correctness

Rendering

[Wann Jensen 1995]

PDEs



[Sawhney & Crane 2020]

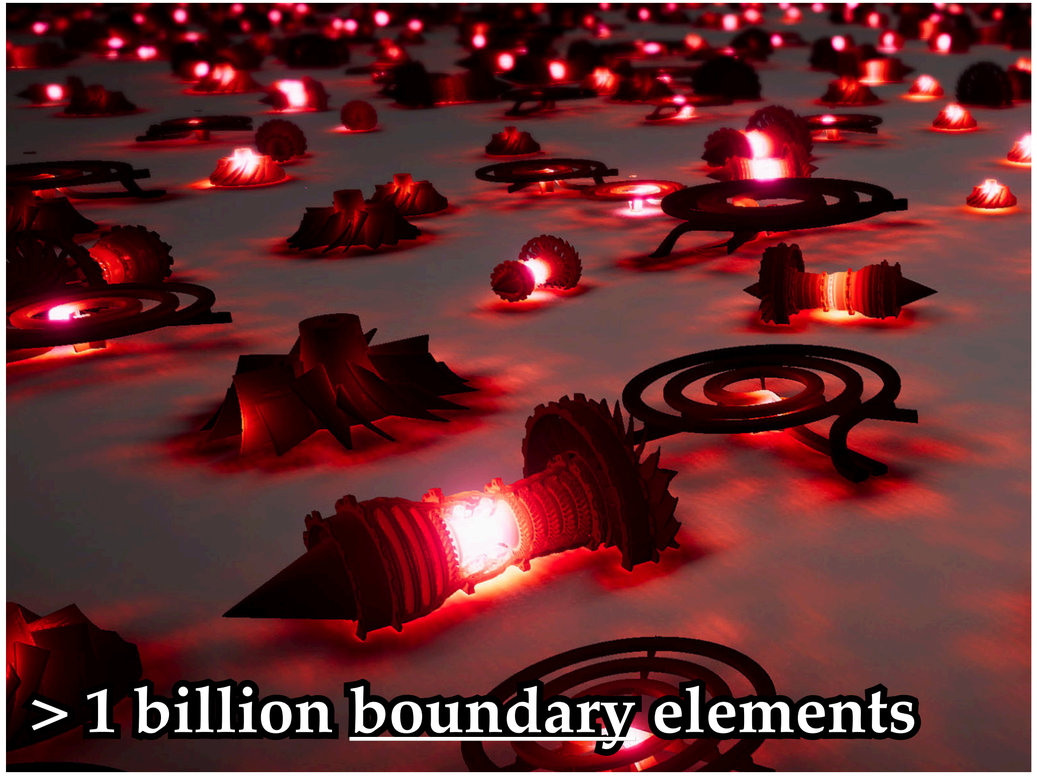
Key idea: as long as equation is well-posed, numerical solution *will* be correct

Benefits of Monte Carlo – Scalability

Rendering

[Georgiev et al 2018]

PDEs



[Sawhney, Seyb, Jarosz, Crane 2022]

Key idea: cost of geometric detail grows like $O(n \log n)$

Benefits of Monte Carlo – Parallelism

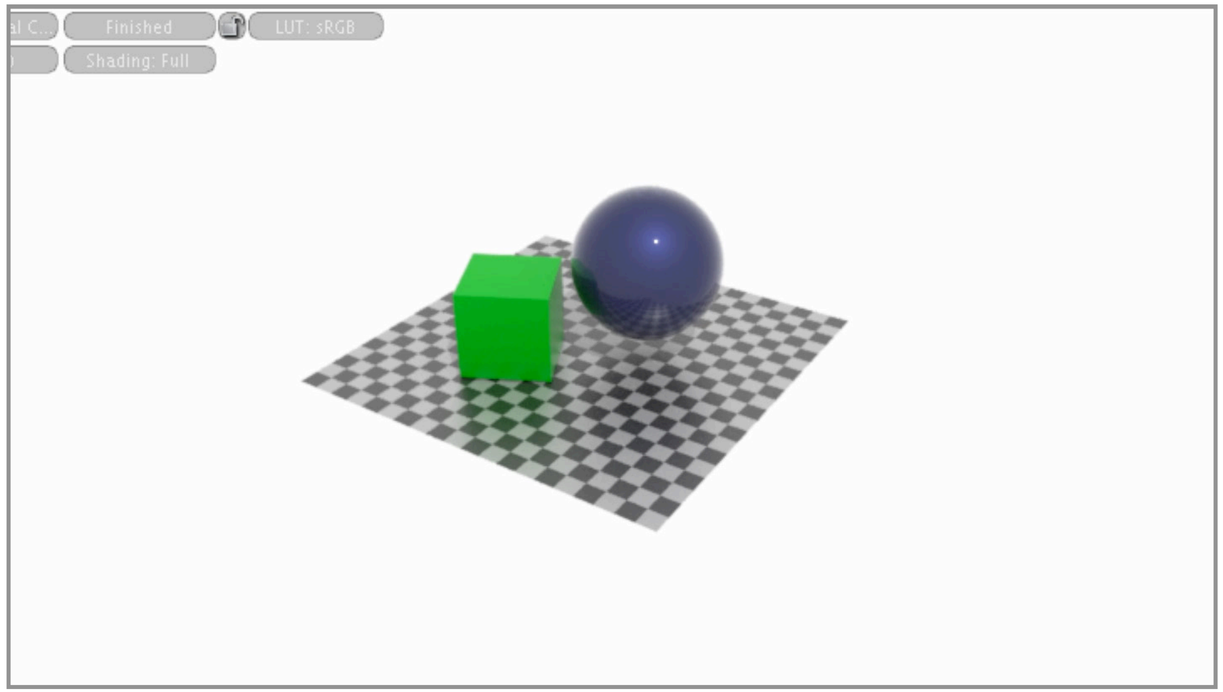
Rendering

PDEs

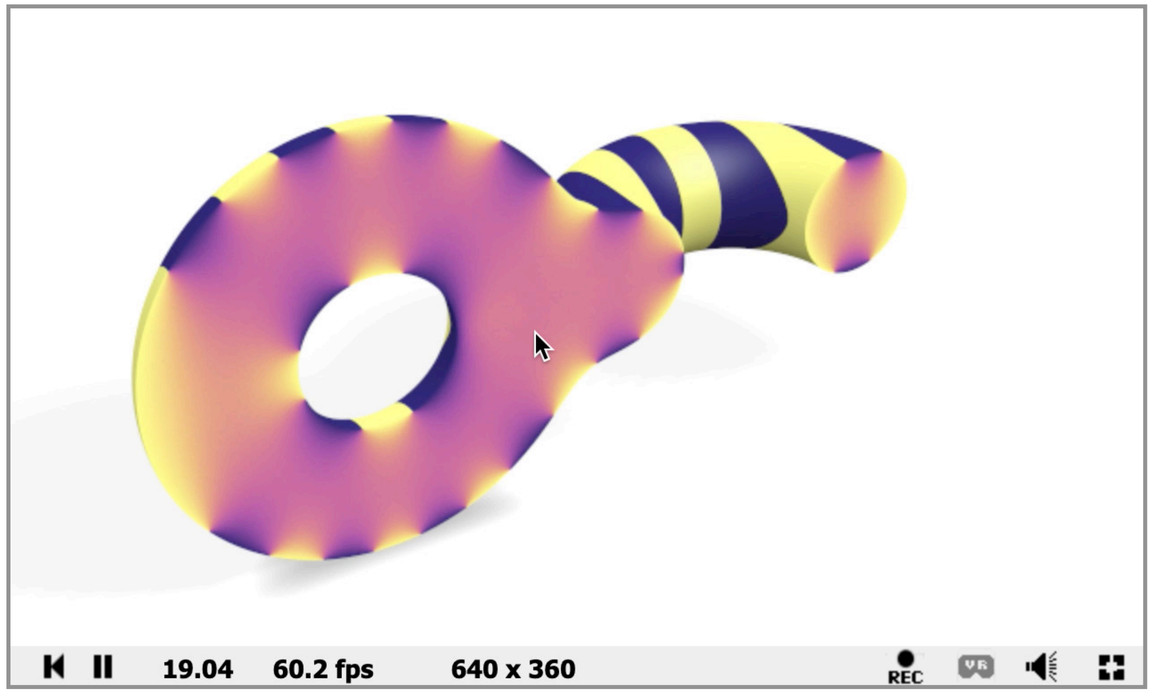
Key idea: just run on N processors, take average of N final estimates

Benefits of Monte Carlo – Progressive

Rendering



PDEs

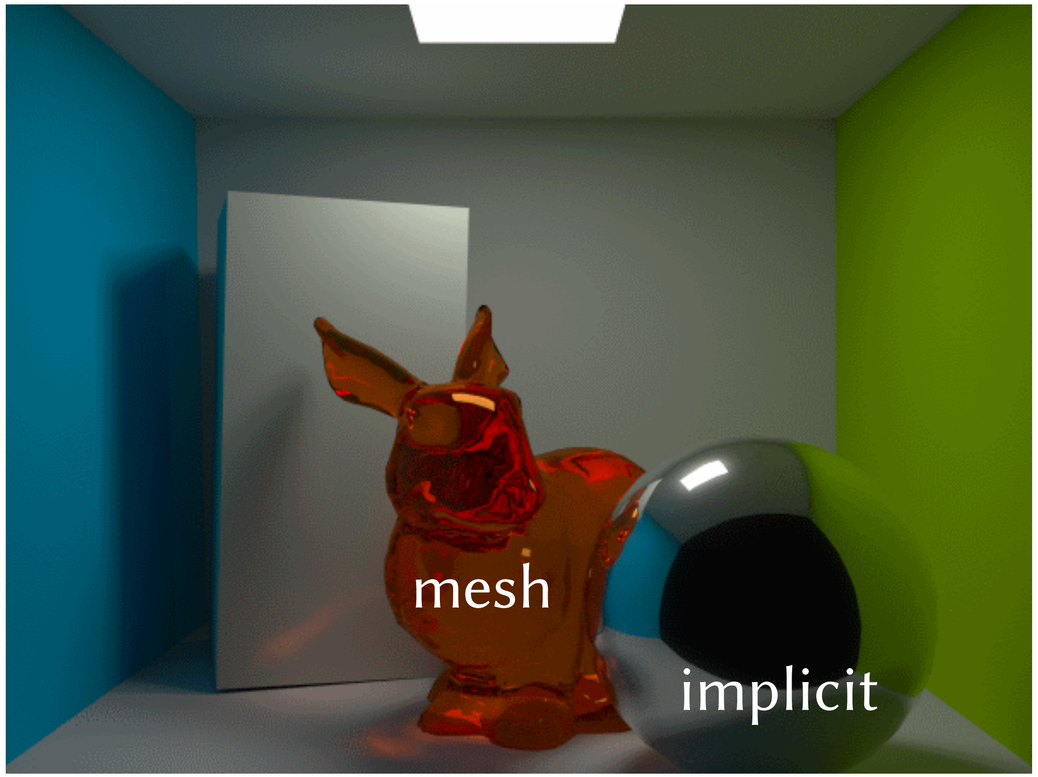


Credit: Ricky Reusser

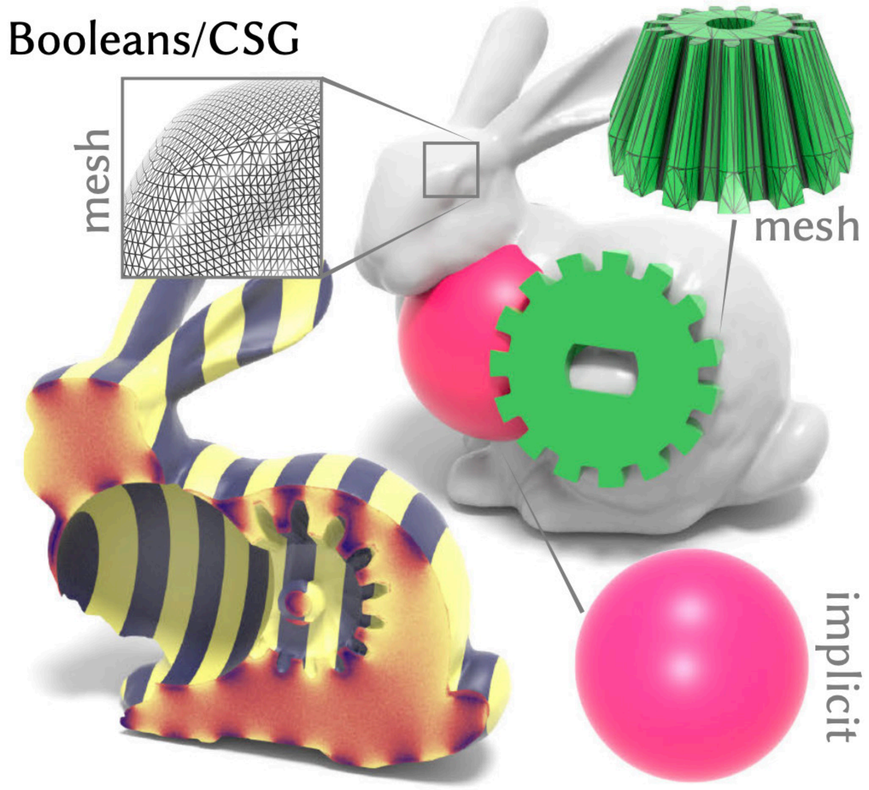
Key idea: fast-but-reliable "preview" enables instant exploration

Benefits of Monte Carlo – Geometric Generality

Rendering



PDEs

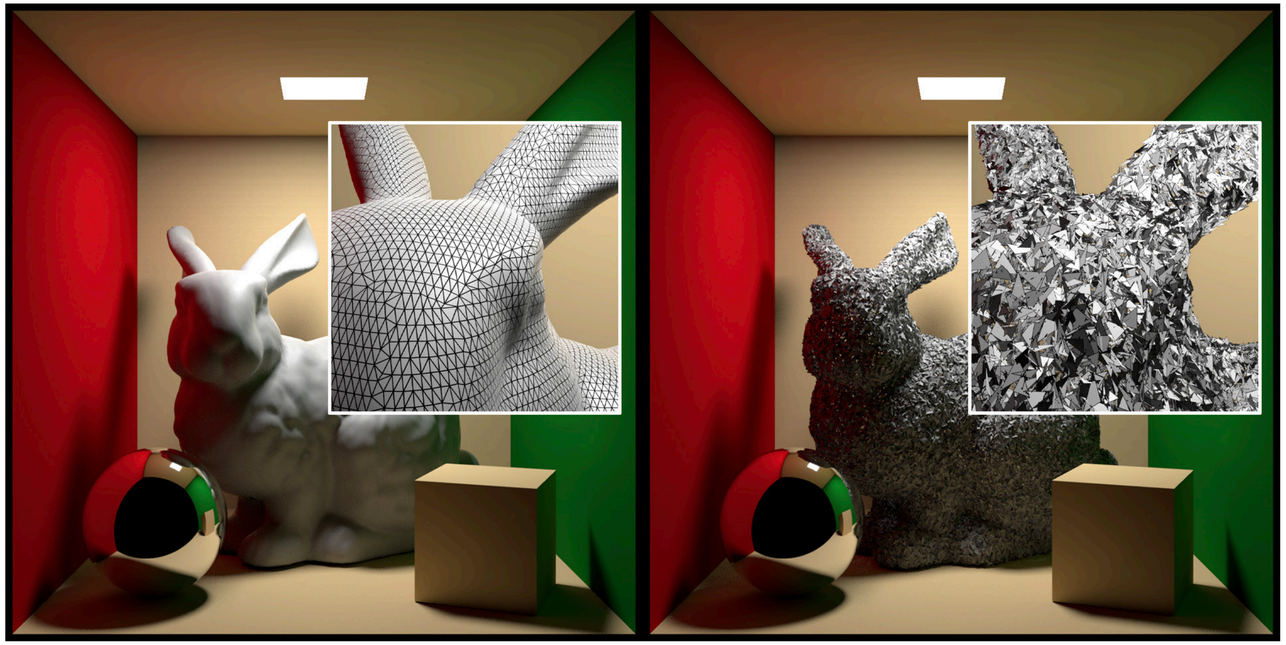


[Sawhney & Crane 2020]

Key idea: can work directly with *heterogeneous* geometry, without conversion

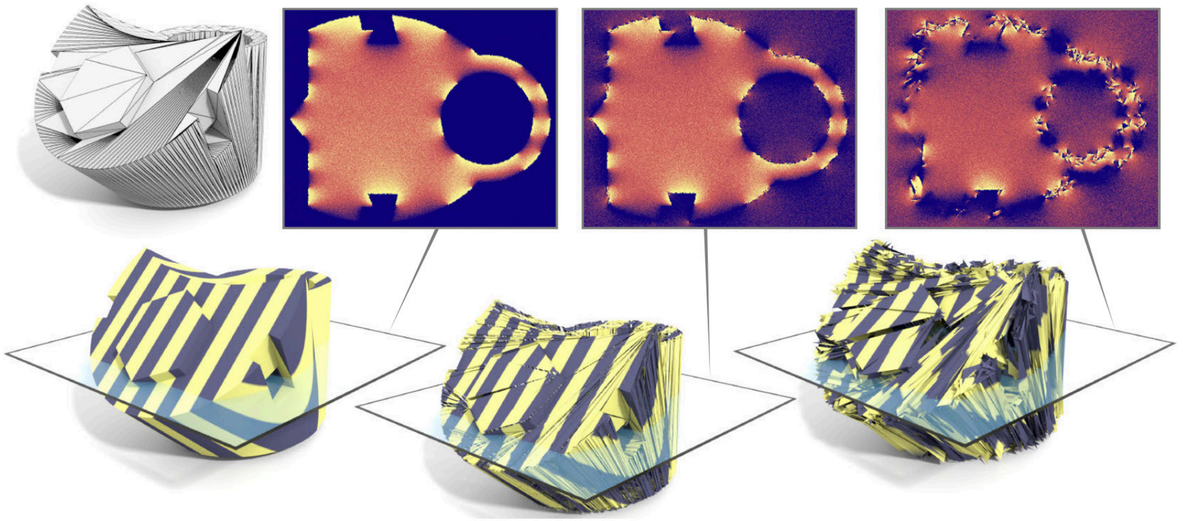
Benefits of Monte Carlo – Robustness

Rendering



Beautiful. Still beautiful.

PDEs

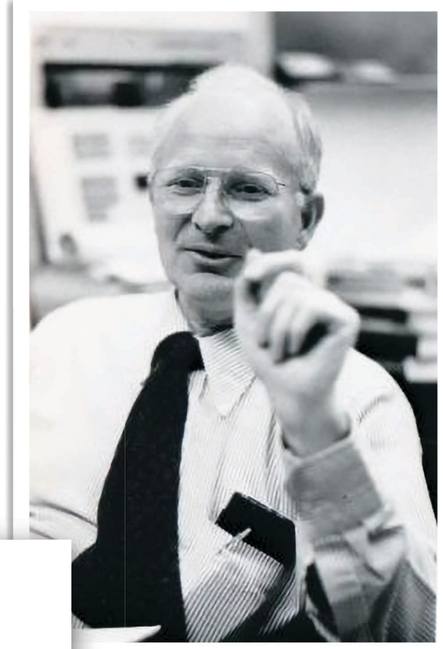


[Sawhney & Crane 2020]

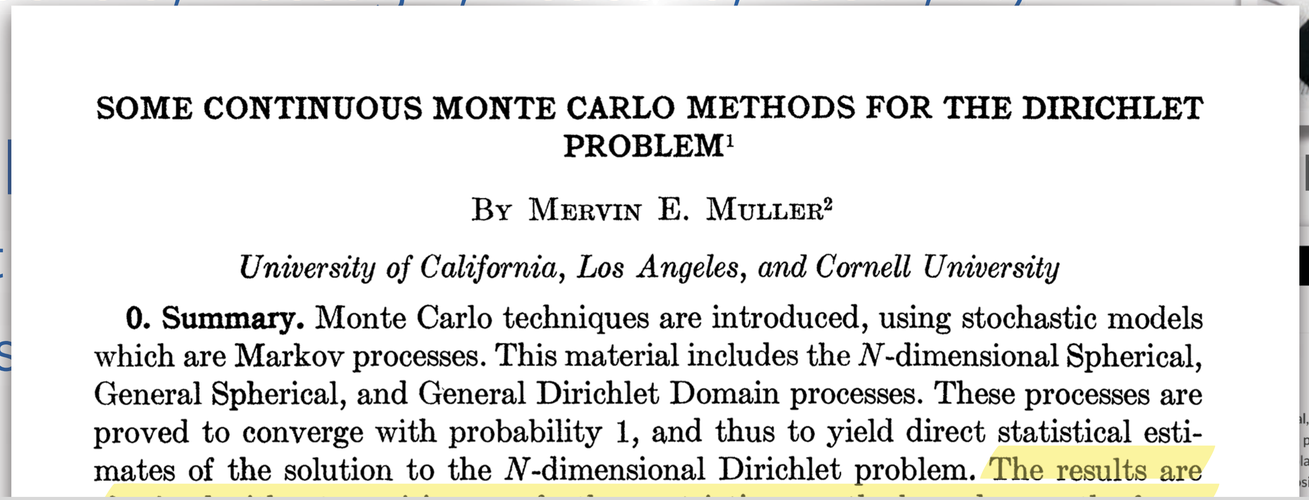
Key idea: solution quality degrades *gracefully*

History of grid-free methods

- **Theory & Algorithms**
  - Brownian motion related to heat & Laplace equations [Einstein 1905]
  - *walk on spheres* [Muller 1956]
  - a.k.a. *floating random walk* [Haji-Sheikh & Sparrow 1966]
  - complexity analysis [Binder & Braverman 2012]
  - extensions (Simonov, Sabelfeld, Mascagni, Deaconu, Booth, …)
- **Applications**
  - integrated circuit design [
  - porous media [Hwang et
  - molecular dynamics [Mas
  - *…not much else!*



Mervin Muller

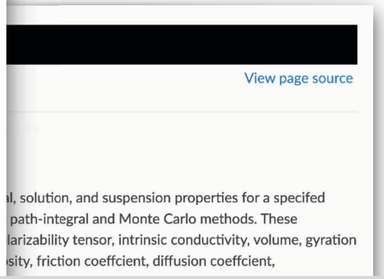


SOME CONTINUOUS MONTE CARLO METHODS FOR THE DIRICHLET PROBLEM[1]

By Mervin E. Muller[2]

University of California, Los Angeles, and Cornell University

0. Summary. Monte Carlo techniques are introduced, using stochastic models which are Markov processes. This material includes the N-dimensional Spherical, General Spherical, and General Dirichlet Domain processes. These processes are proved to converge with probability 1, and thus to yield direct statistical estimates of the solution to the N-dimensional Dirichlet problem. The results are



View page source

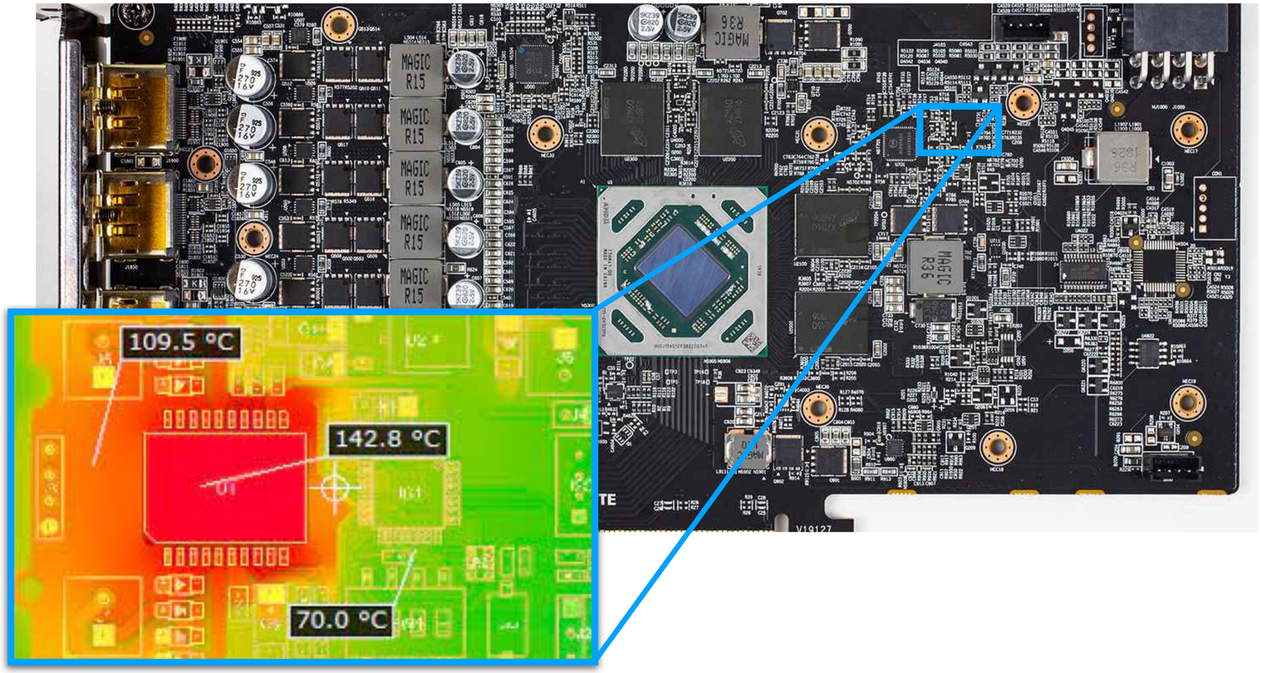
l, solution, and suspension properties for a specifed
path-integral and Monte Carlo methods. These
larizability tensor, intrinsic conductivity, volume, gyration
sity, friction coeffcient, diffusion coeffcient,

Main challenge: so far, applies to a limited class of PDEs!

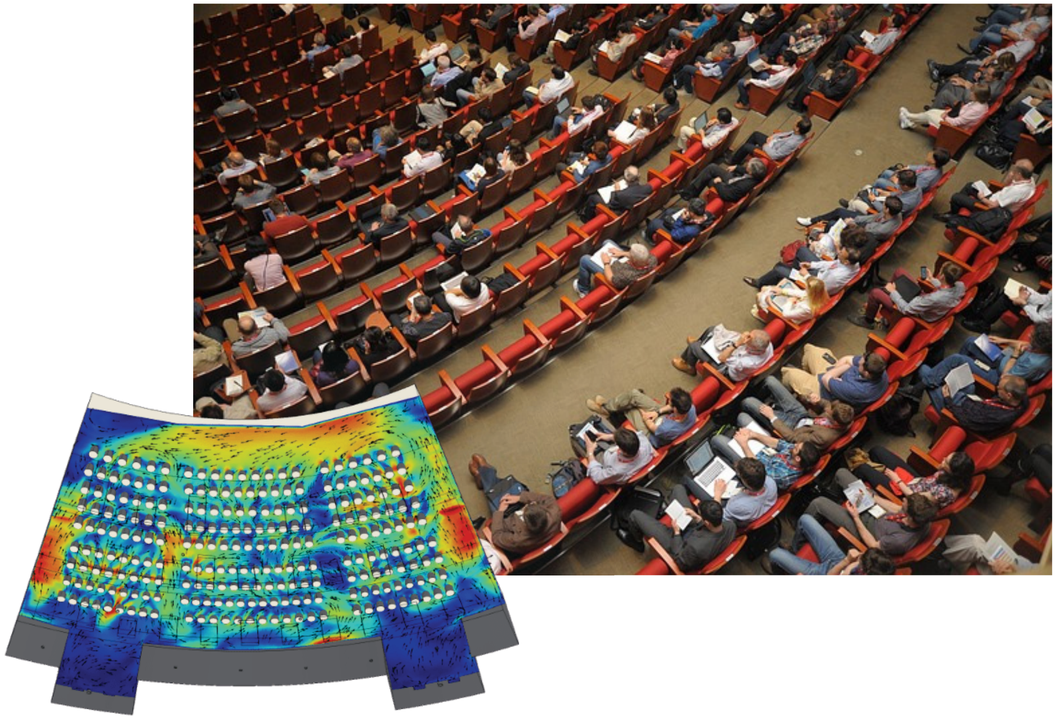
PART 3:
WoS for Poisson Equation

Motivation for source terms

thermal analysis of PCBs



thermal management in building design



Roadmap for solving Poisson PDE

differential form

Poisson PDE

Find u such that

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

integral representation

Green's integral

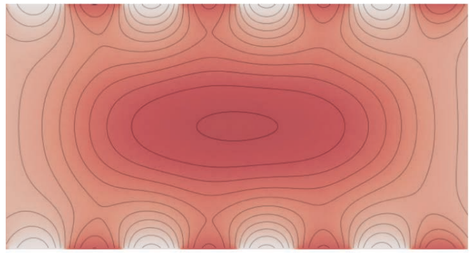
$$\int_{B(x)} G(x,z) f(z)\, dz$$

Monte Carlo estimator

Walk on Spheres+

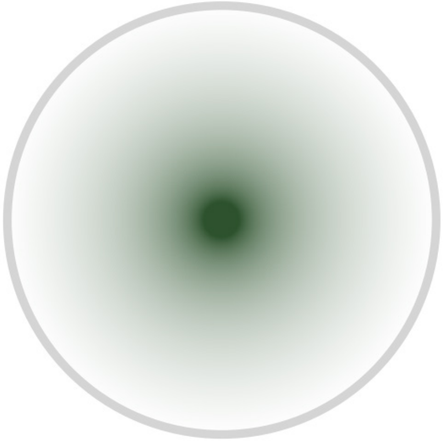
$$\hat{u}(x_i) = \begin{cases} \hat{u}(x_{i+1}) + |B(x_i)|\, G(x_i, x_{i+1}) f(x_{i+1}) \\ g(x_{i+1} \end{cases}$$

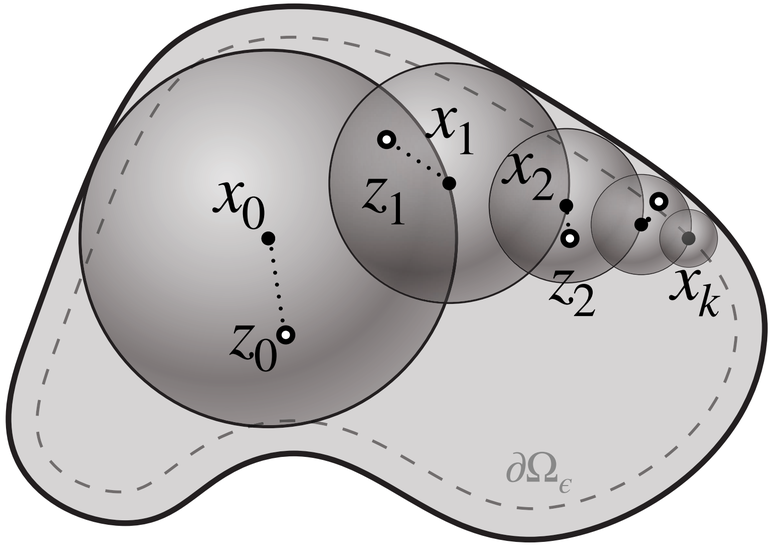
$u(x)$



$f(x)$

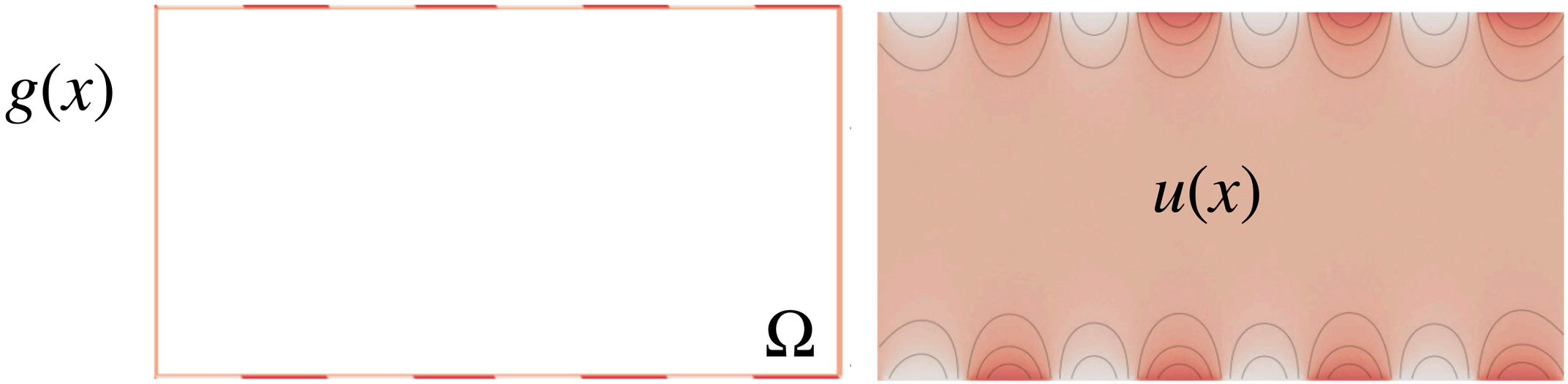
$g(x)$





Dirichlet boundary condition

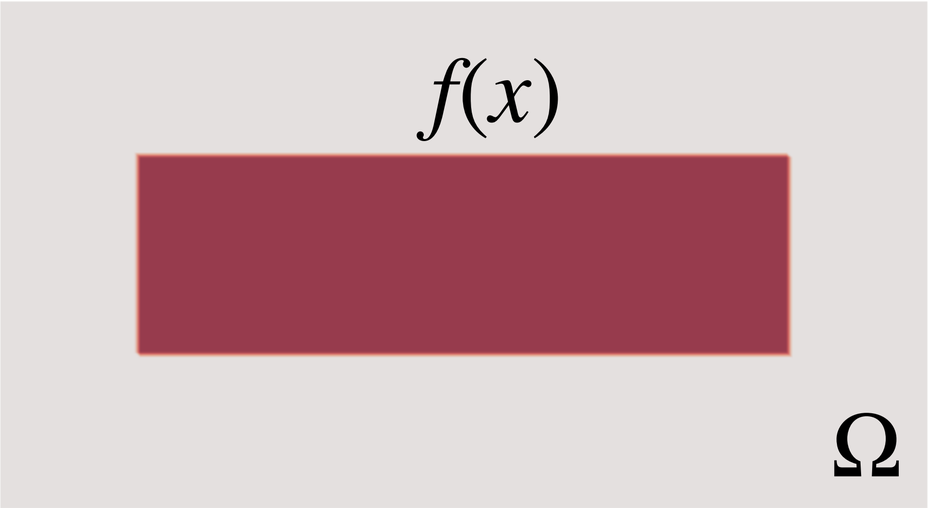
$$\Delta u = 0 \quad \text{on } \Omega$$
$$u = g \quad \text{on } \partial\Omega$$

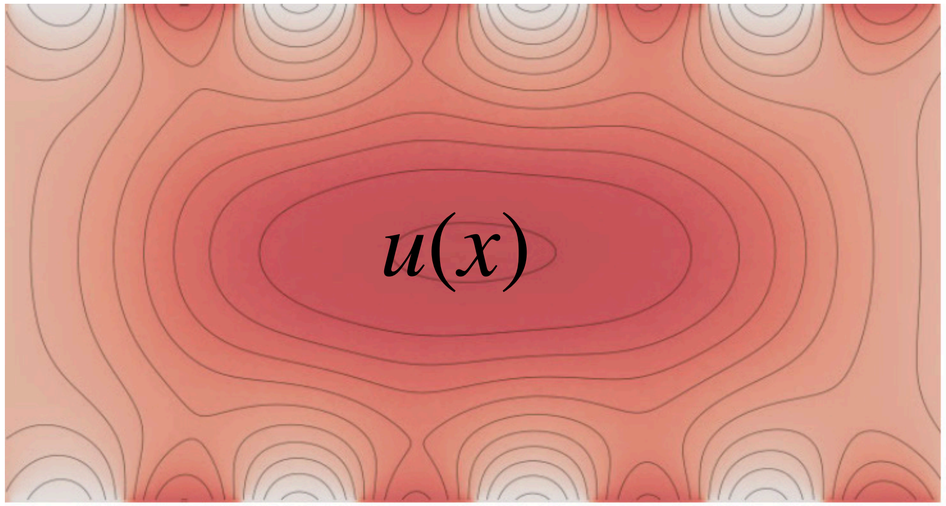


Intuition: temperature prescribed on the boundary

Poisson equation

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





Intuition: adds additional background temperature

Green's function

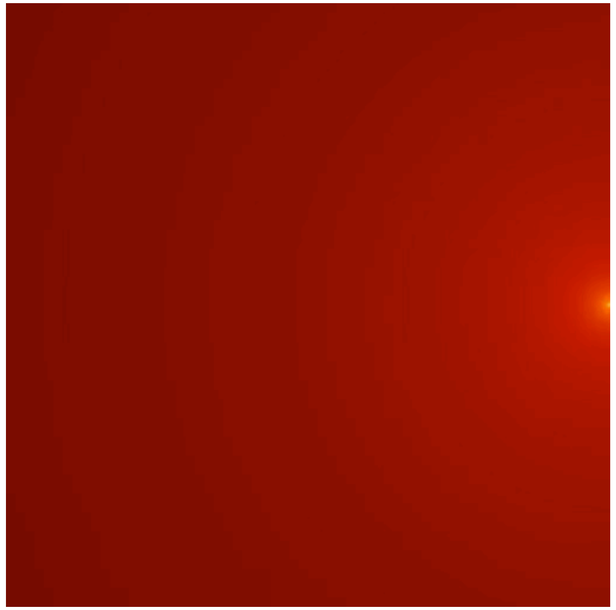
$$\Delta G(x, z) = \delta_z(x)$$

heat injected at single point

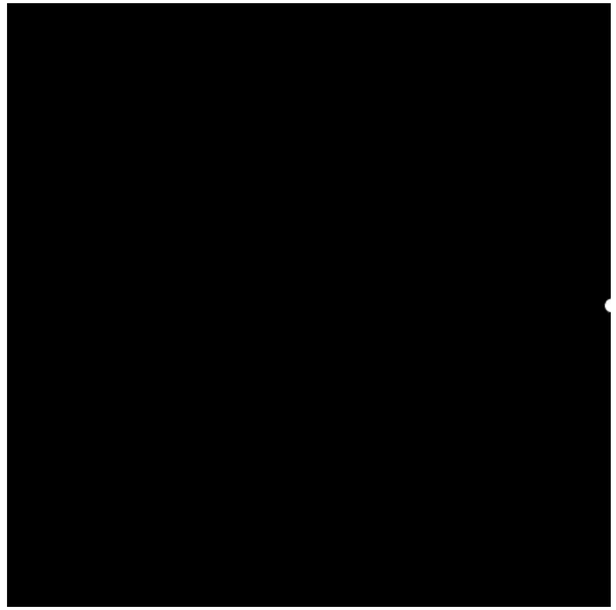
free-space Green's function

$$G^{2D}(x, z) = \frac{-\ln(|x - z|)}{2\pi}$$

$$G^{3D}(x, z) = \frac{1}{4\pi|z - x|}$$



$G(x, z)$



$\delta_z(x)$

Green's function

$$\Delta G_B(x,z) = \delta_z(x) \quad \text{on } B$$

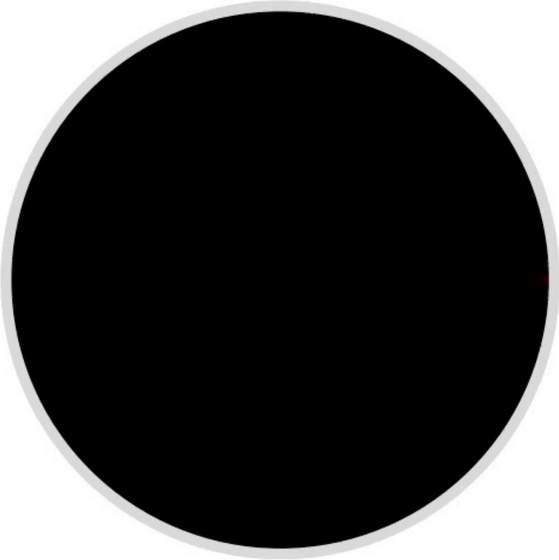
$$G_B(x,z) = 0 \quad \text{on } \partial B$$

zero temperature at boundary

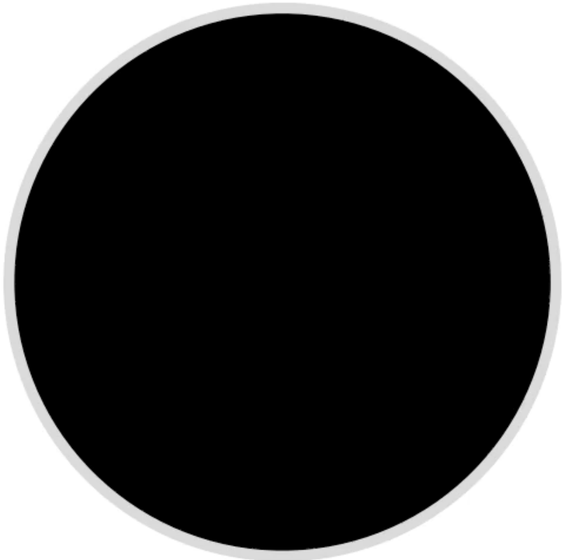
Green's function on a ball

$$G_B^{2D}(x,z) = \frac{1}{2\pi} \ln\left(\frac{R^2 - \langle x,z\rangle}{R\,|z-x|}\right)$$

$$G_B^{3D}(x,z) = \frac{1}{4\pi}\left(\frac{1}{4\,|x-z|} - \frac{R}{R^2 - \langle x,z\rangle}\right)$$



$G_B(x,z)$



$\delta_z(x)$

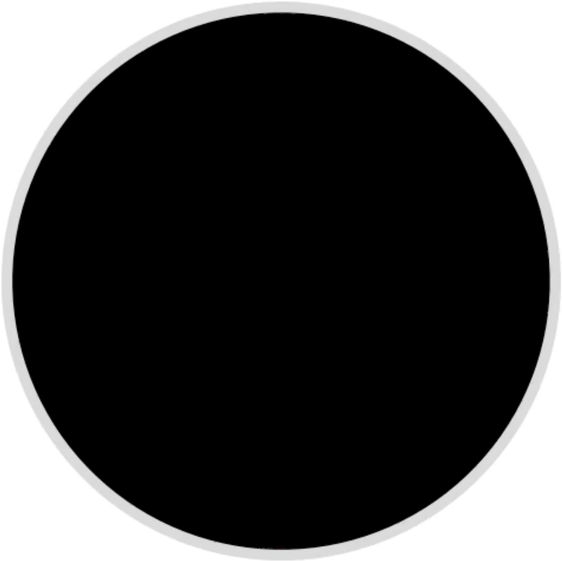
Sources are additive

Sources are additive, so we can sum together the Green's function to model multiple Dirac deltas

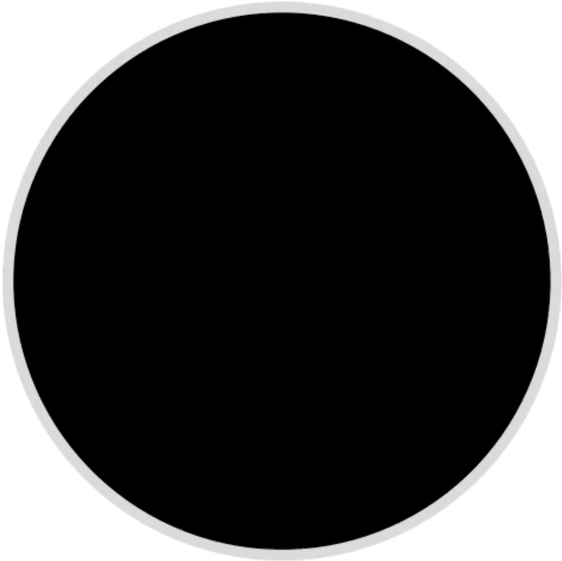
$$\Delta w(x) = \delta_{z_1}(x) + \delta_{z_2}(x) \quad \text{on } B$$
$$w(x) = 0 \quad \text{on } \partial B$$

$$w(x) = G_B(x, z_1) + G_B(x, z_2)$$

$w(x)$



$\delta_{z1}(x) + \delta_{z2}(x)$



Green's integral on a ball

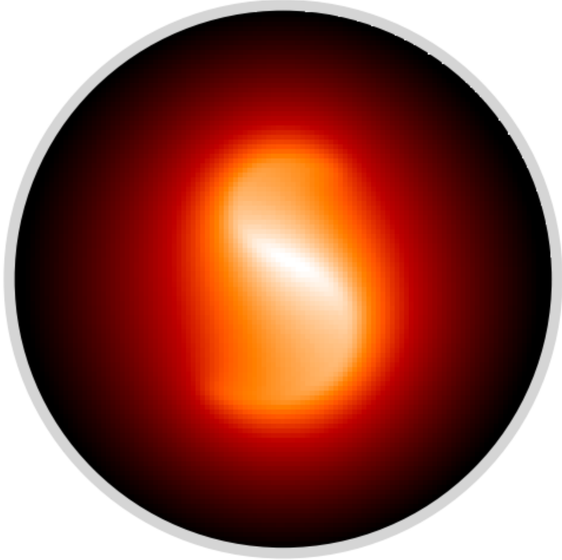
By convolving the Green's integral with a source term we obtain the source contribution over the ball.

$$\Delta w(x) = f(x) \quad \text{on } B$$
$$w(x) = 0 \quad \text{on } \partial B$$

Green's integral

$$w(x) = \int_B G_B(x, z) f(z)\ dz$$

$w(x)$



$f(z)$

Generalized mean value integral

The mean value integral is generalized by considering the source contribution within the ball centered at x.

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

generalized mean value integral

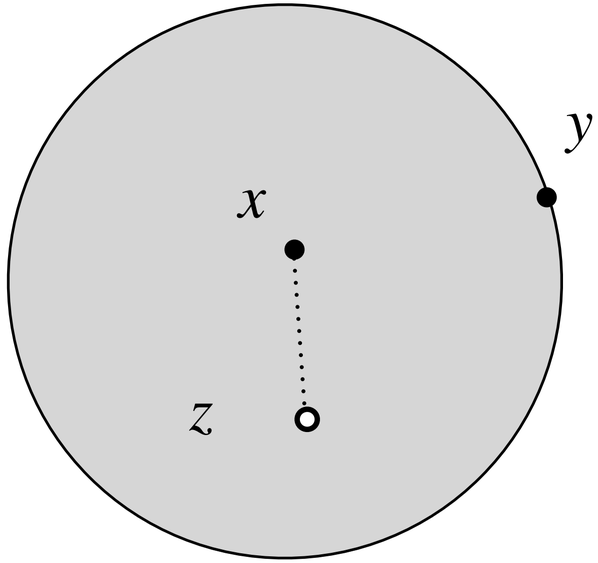
$$u(x) = \frac{1}{|\partial B(x)|}\int_{\partial B(x)} u(y)\ dy + \int_{B(x)} G(x,z)f(z)\ dz$$

boundary source

generalized mean value estimator

$$\hat{u}(x_i) = u(y) + |B(x)|\,G(x,z)f(z)$$

$$y \sim \mathcal{U}_{\partial B(x)} \qquad z \sim \mathcal{U}_{B(x)}$$



Walk on spheres with source term

The mean value integral can be generalized using the Green's integral to accounts for source terms

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

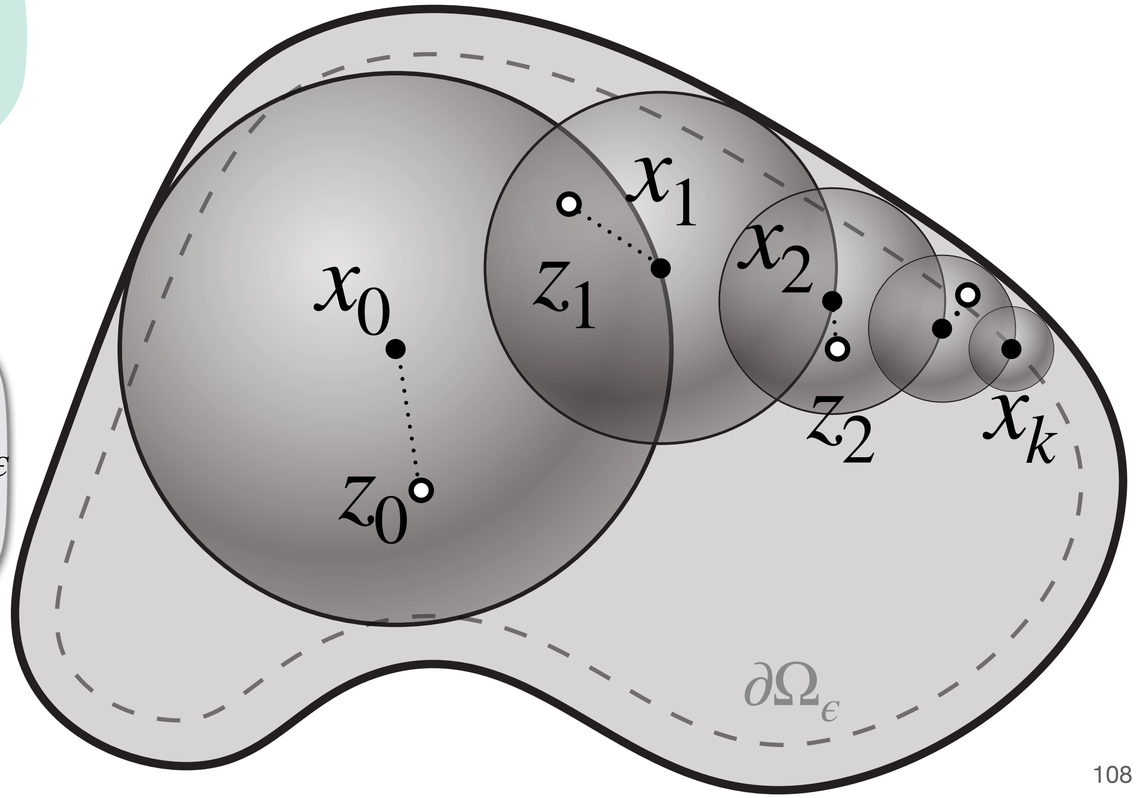
generalized mean value integral

$$u(x) = \underbrace{\frac{1}{|\partial B(x)|}\int_{\partial B(x)} u(y)\, dy}_{\text{boundary}} + \underbrace{\int_{B(x)} G(x,z) f(z)\, dz}_{\text{source}}$$

walk on spheres estimator

$$\hat{u}(x_i) = \begin{cases} \hat{u}(x_{i+1}) + |B(x_i)|\, G(x_i, z_i) f(z_i) & \text{if } x_{i+1} \notin \partial\Omega_\epsilon \\ g(x_{i+1}) & \text{otherwise} \end{cases}$$

$$x_{i+1} \sim \mathcal{U}_{\partial B(x)} \qquad z_i \sim \mathcal{U}_{B(x_i)}$$



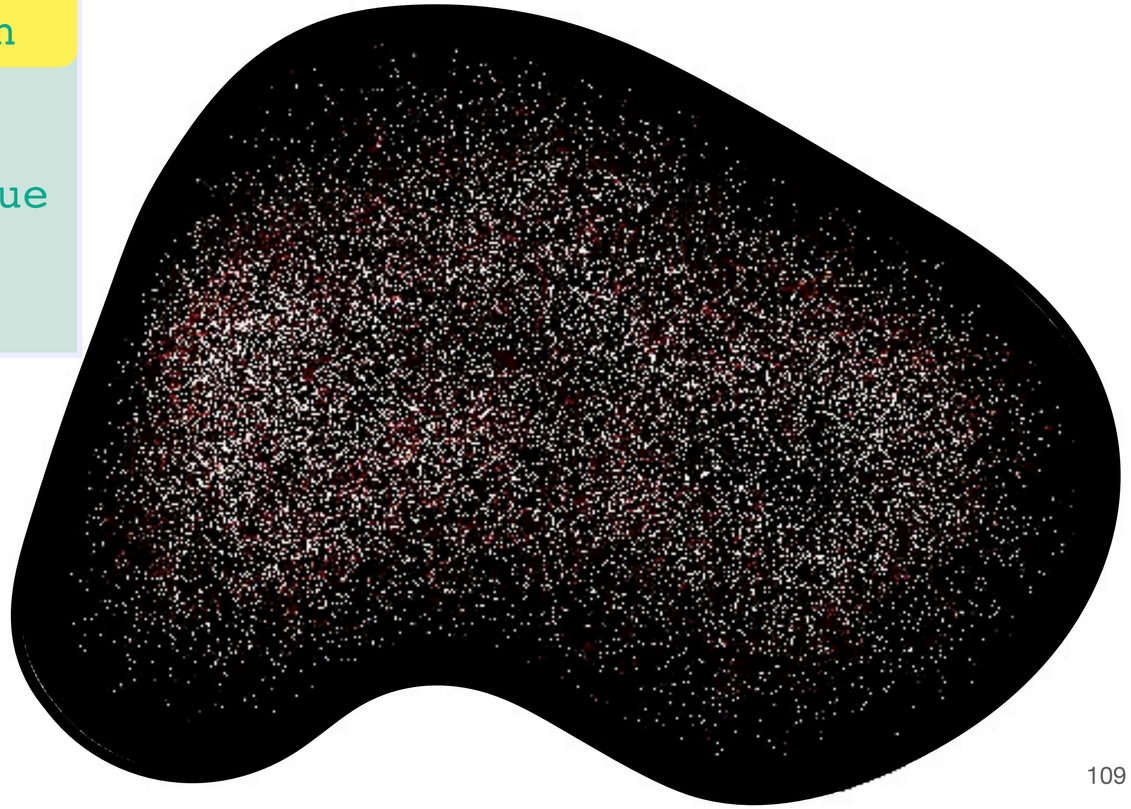
Walk on spheres with source term

```
u = 0 // solution estimate
for i=1,..,nWalks {
   x = x0 // start a new walk
   do {
      // move to random point on biggest empty sphere
      r = distance(x,∂Ω)
      z = randomBall(x,r) // random point inside
      u += π*r*r*G(x,z,r)*f(z) // source contribution
      x = randomSphere(x,r)
   } while(r > ε) // close enough!
   u += g(closestPoint(x,∂Ω)) // sample boundary value
}
return u/nWalks // return average boundary value
```

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

walk on spheres estimator

$$\hat{u}(x_i) = \begin{cases} \hat{u}(x_{i+1}) + |B(x_i)|\, G(x_i, z_i) f(z_i) & \text{if } x_{i+1} \notin \partial\Omega_\epsilon \\ g(x_{i+1}) & \text{otherwise} \end{cases}$$



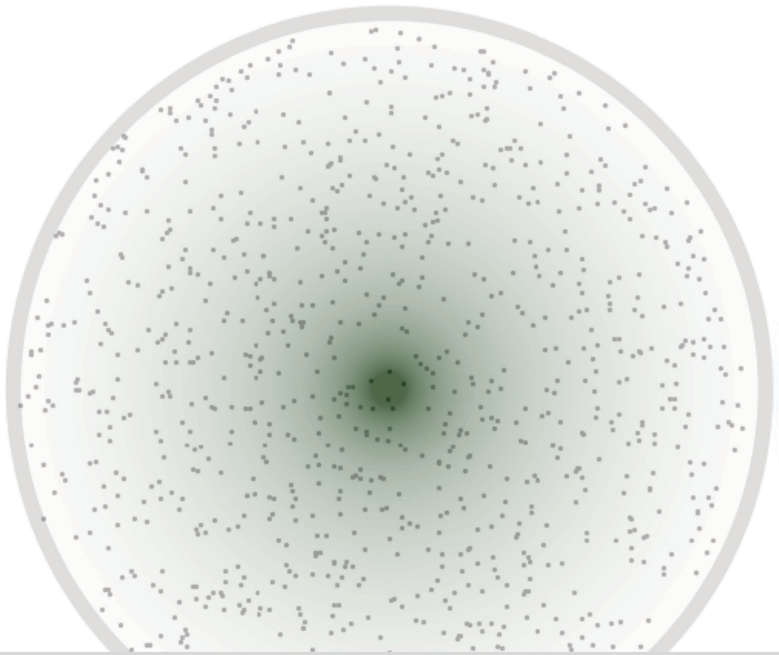
Uniform sampling limitations

$$I = \int_{B(x)} G(x, z) f(z)\ dz$$

$$\hat{I}_{\text{uniform}} = \frac{|B(x)|}{n} \sum_{i=1}^{n} G(x, z_i) f(z_i)$$

uniform samples

Green's function $G(x, z)$

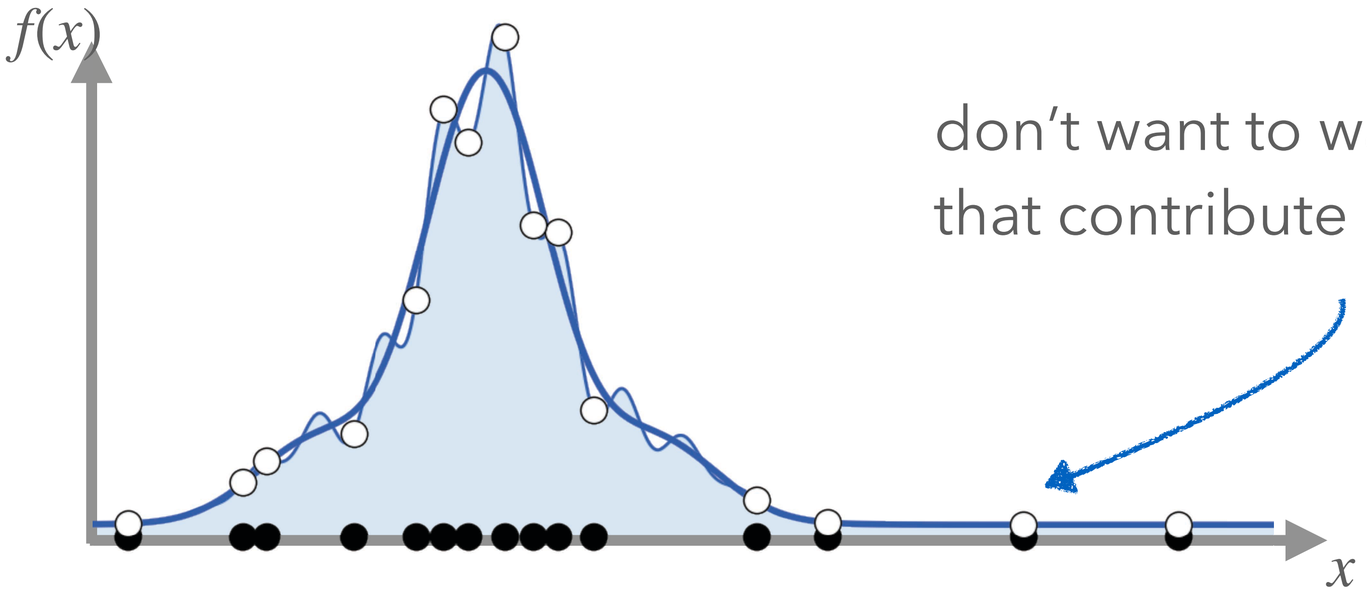


source function $f(z)$

Observation: most uniform samples don't contribute much…

Importance sampling to the rescue

Idea: concentrate samples where the integrand is large



don't want to waste time on points that contribute little to integral

$$I = \int_a^b f(x)\ dx \qquad \hat{I}_{Importance} = \frac{1}{n}\sum_{i=1}^{n} \frac{f(X_i)}{p(X_i)}$$

draw more samples where p is large

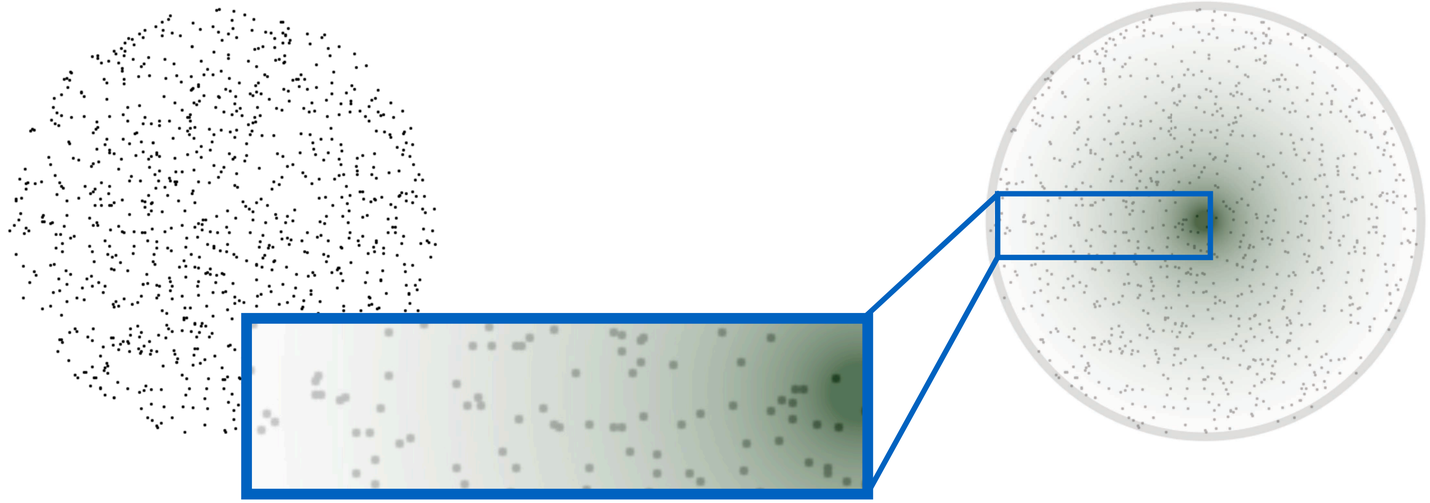
$$X_i \sim p$$

corrects for under / over sampling of uniform distribution

Sampling distributions: uniform vs Green's function

uniform samples

Green's function $G(x, z)$



importance samples

Green's function $G(x, z)$

key: importance sampling improves efficiency

Variance of basic Monte Carlo estimator

How do we quantify a "better" sampling PDF $p(z)$?

variance

$$V[X] = E[(X - E[X])^2]$$

$$V\left[\hat{I}_{\text{importance}}(x)\right] = V\left[\frac{1}{n}\sum_{i=1}^{n}\frac{G(x, Z_i)f(Z_i)}{p(Z_i)}\right]$$

definition of $\hat{I}_{\text{importance}}$

$$= \frac{1}{n^2}V\left[\sum_{i=1}^{n}\frac{G(x, Z_i)f(Z_i)}{p(Z_i)}\right]$$

homogeneity $V[aX] = a^2V[X]$

$$= \frac{1}{n^2}\sum_{i=1}^{n}V\left[\frac{G(x, Z_i)f(Z_i)}{p(Z_i)}\right]$$

independence of Z_i

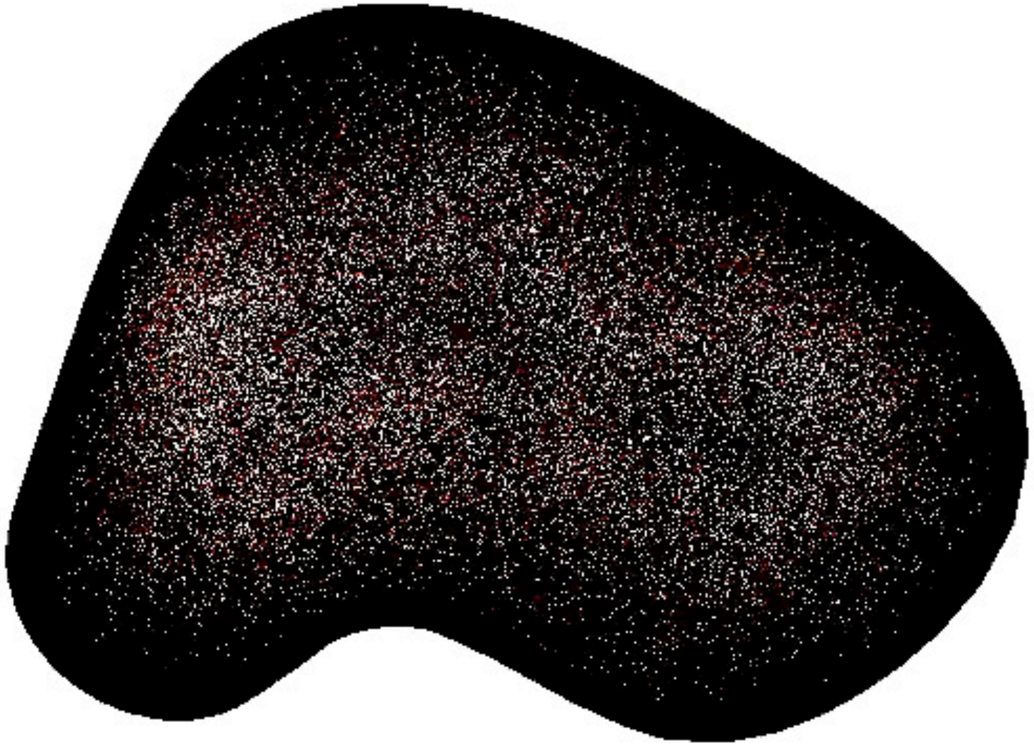
$$= \frac{1}{n}V\left[\frac{G(x, Z)f(Z)}{p(Z)}\right]$$

Lowest variance achieved by making

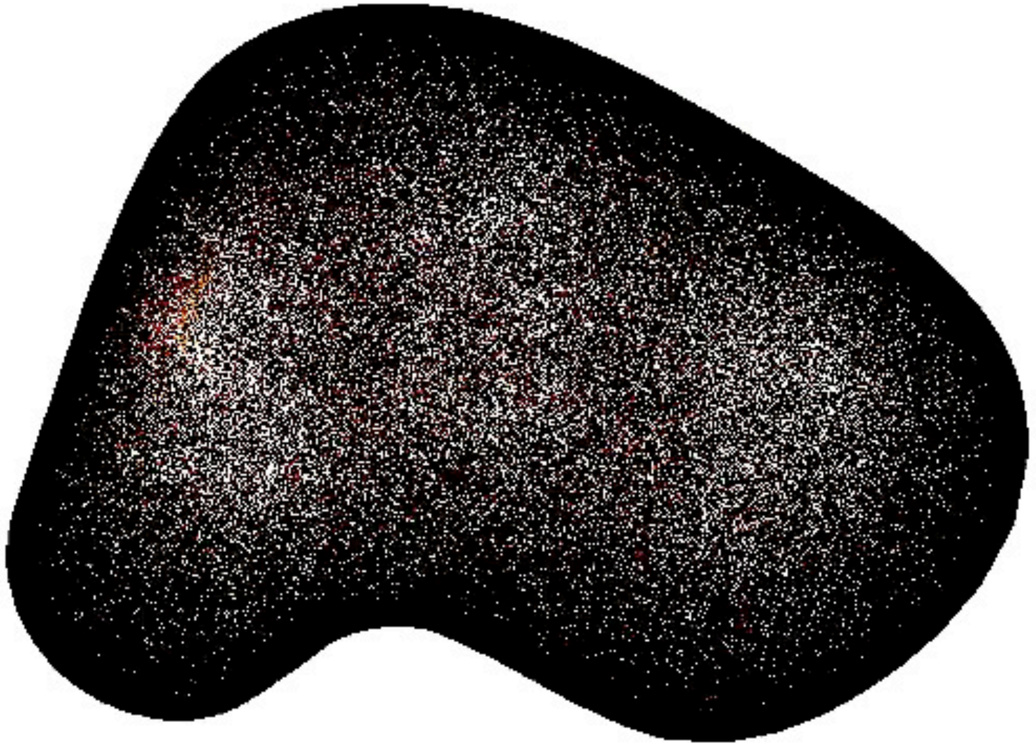
$$p(z) \propto G(x, z)f(z)$$

Walk on spheres with and without importance sampling

uniform sampling

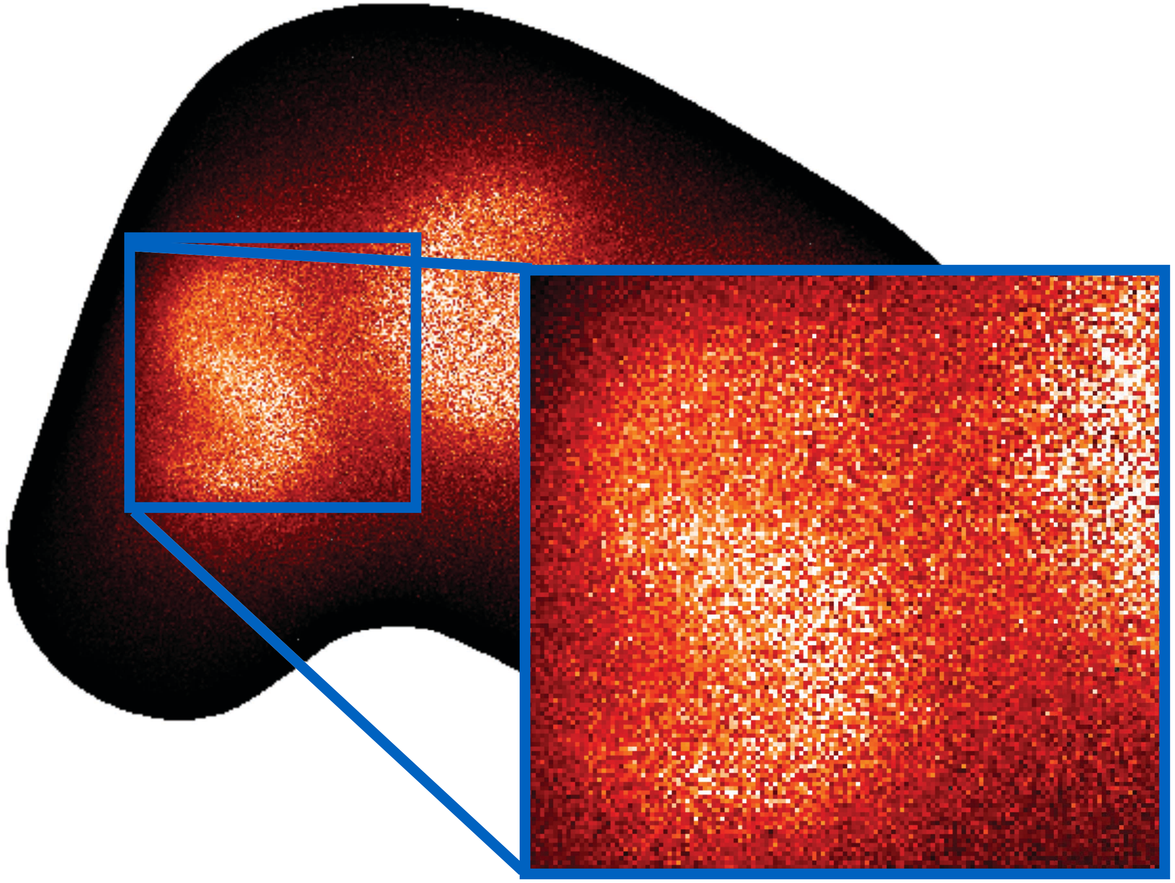


importance sampling Green's function

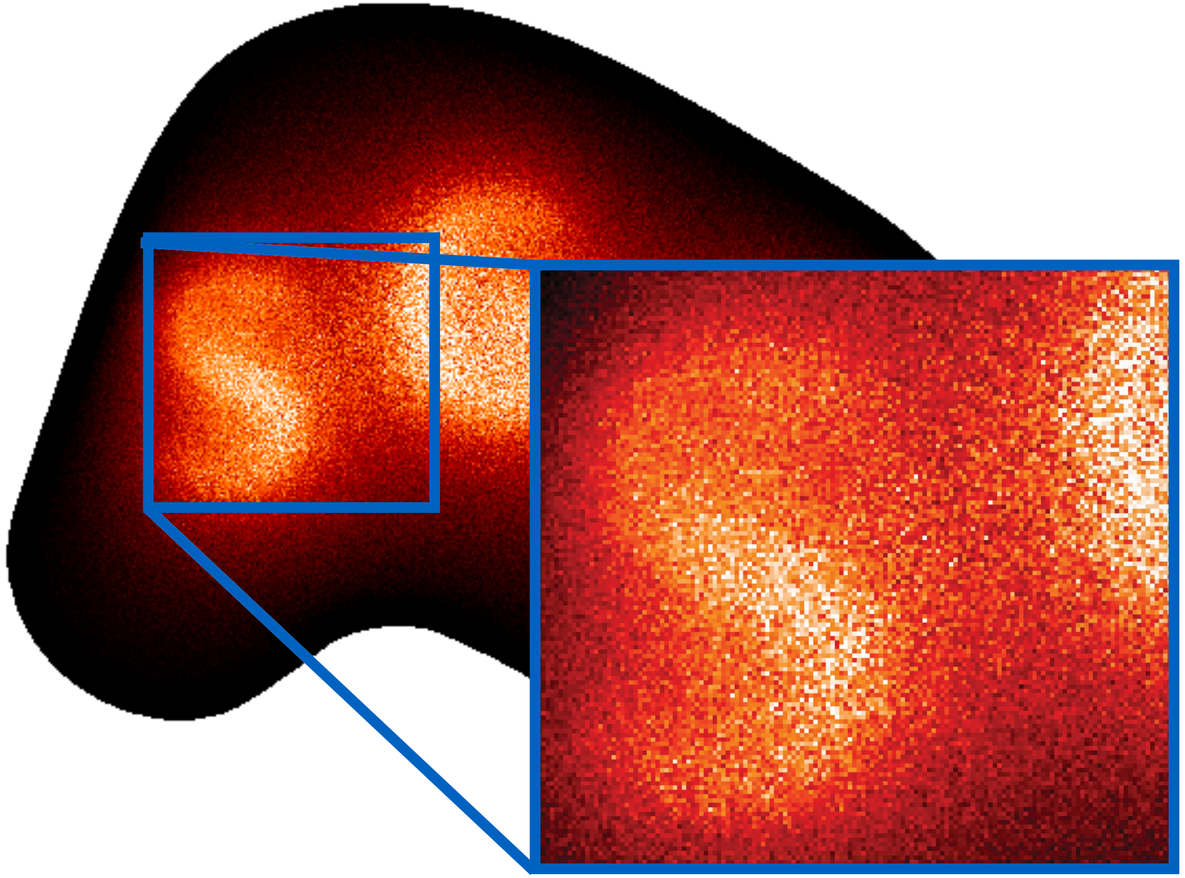


Walk on spheres with and without importance sampling

uniform sampling

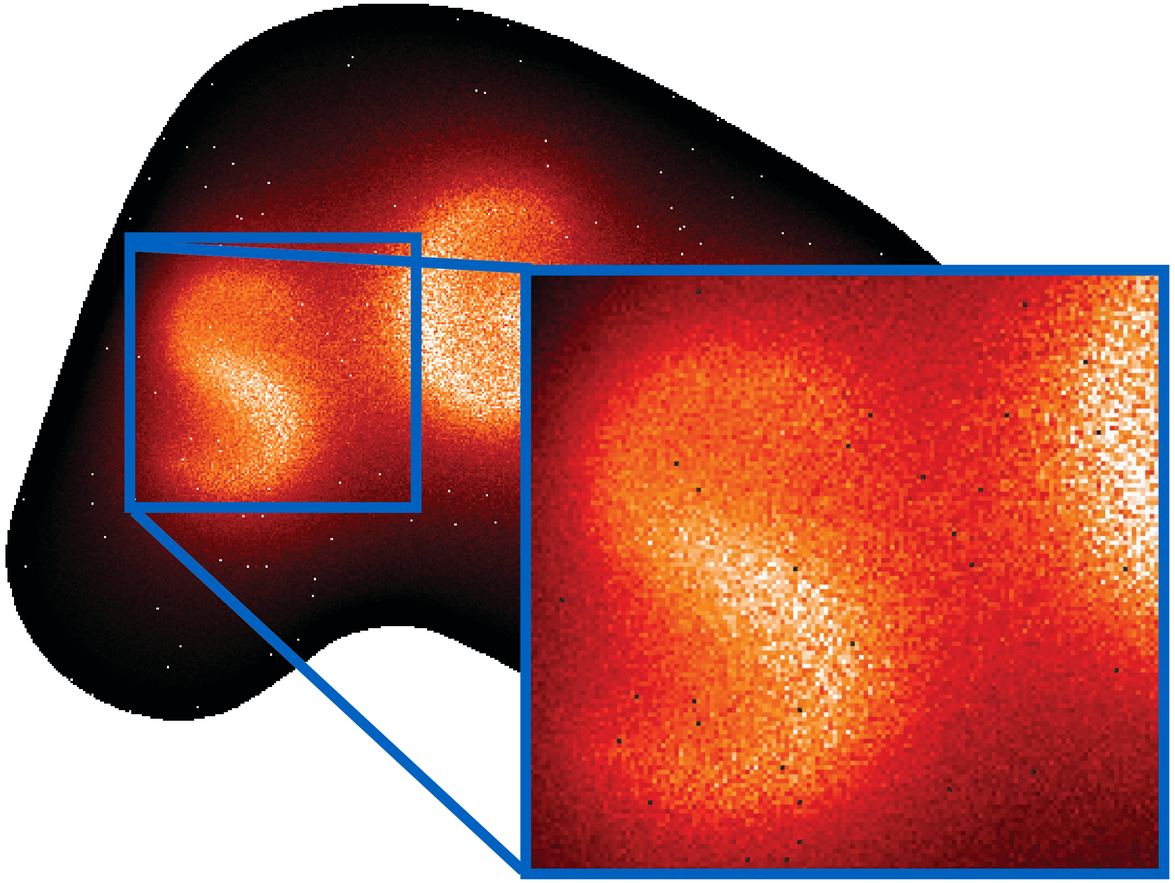


importance sampling Green's function

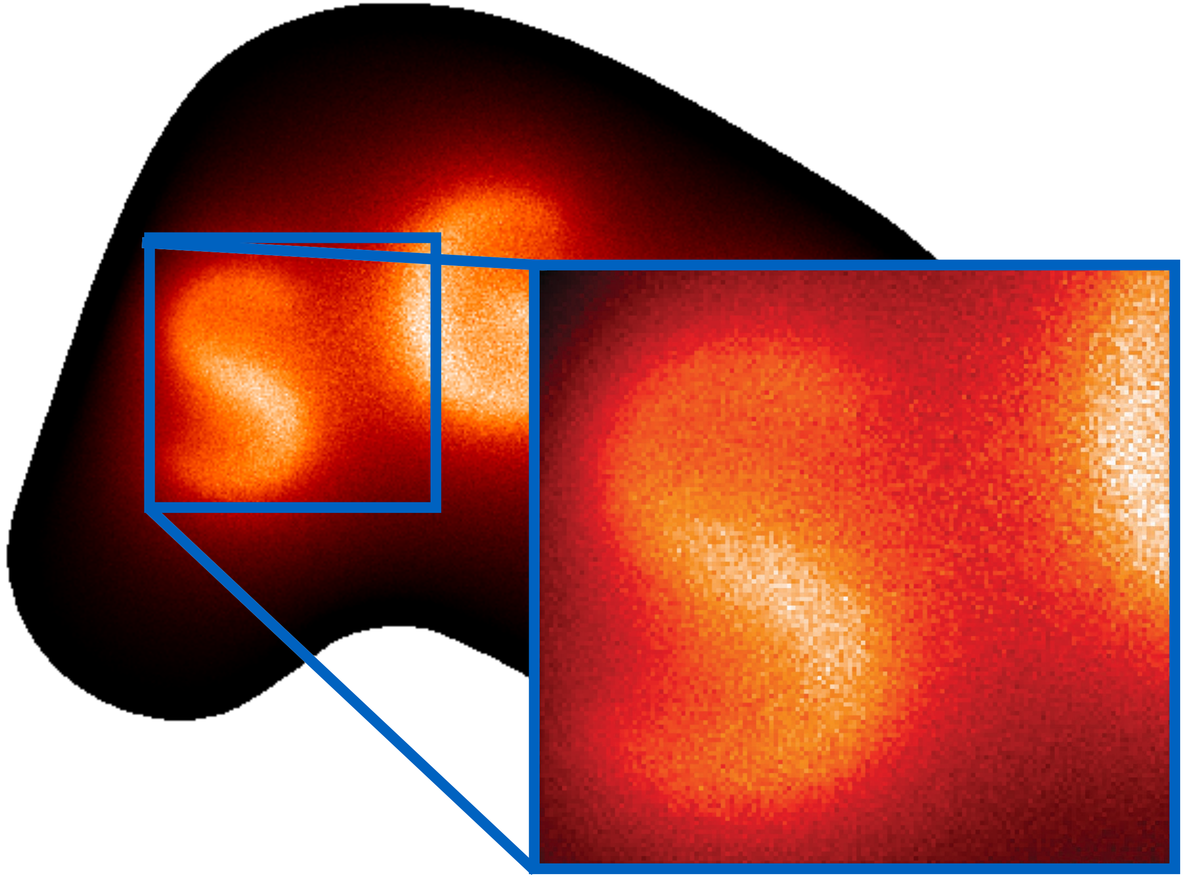


Walk on spheres with and without importance sampling

uniform sampling



importance sampling Green's function



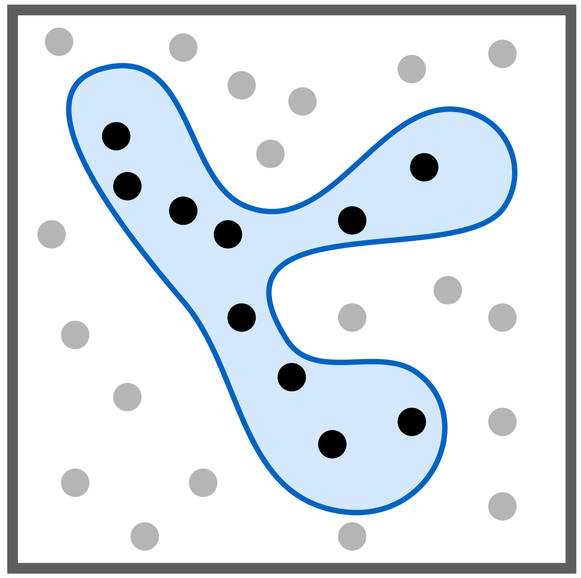
Importance sampling additional terms

uniform samples

source samples $f(z)$

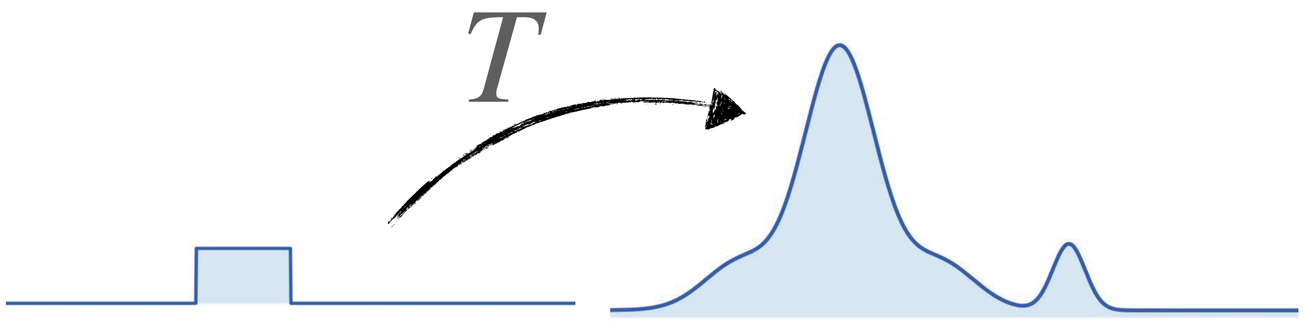
How do we actually sample points?

rejection sampling



Sample from a known distribution,
throw out samples proportionally
to target function.

warping samples



Sample from a known distribution,
warp samples to target.
(e.g. inverse CDF transform)

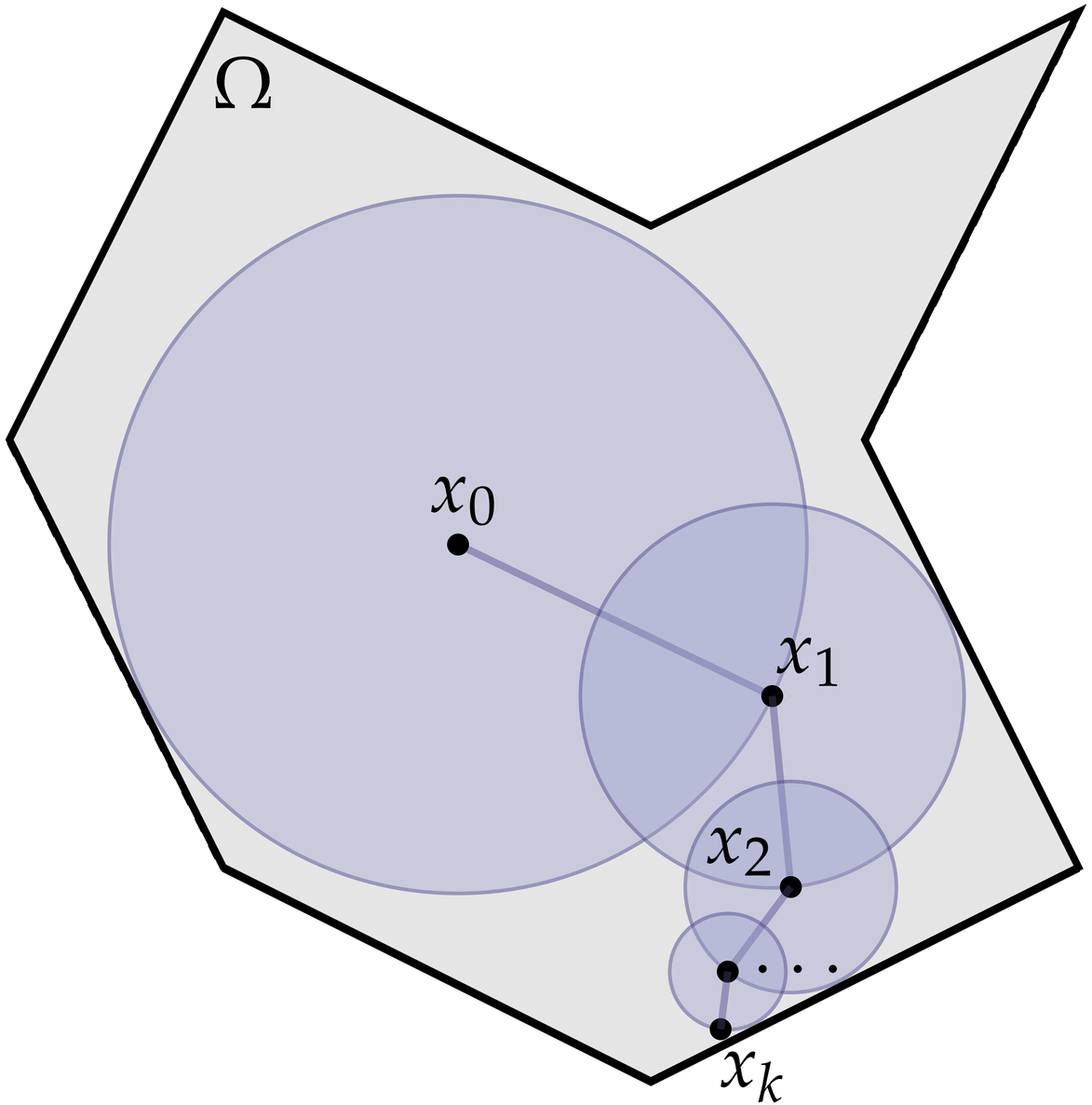
Combining sampling strategies

- In practice, may have more than one importance sampling strategy that seems promising. Which one should you use?
- **Metaphor:** Suppose I'm a soccer goalie, and know my opponent will either shoot left or shoot right–but no great way to predict which one
- Want "robust" strategy that works well no matter what happens

multiple importance sampling provides principled approach to mixture distributions

PART 4:
WoS for Neumann Boundary Conditions

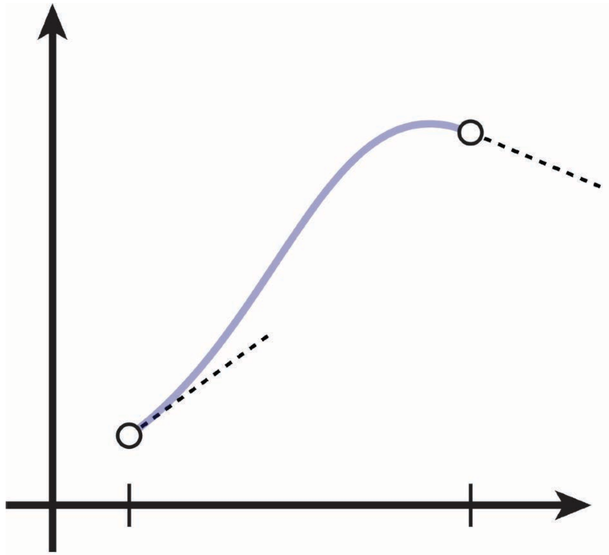
WoS solves PDEs with Dirichlet boundary conditions



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

Laplace eq.

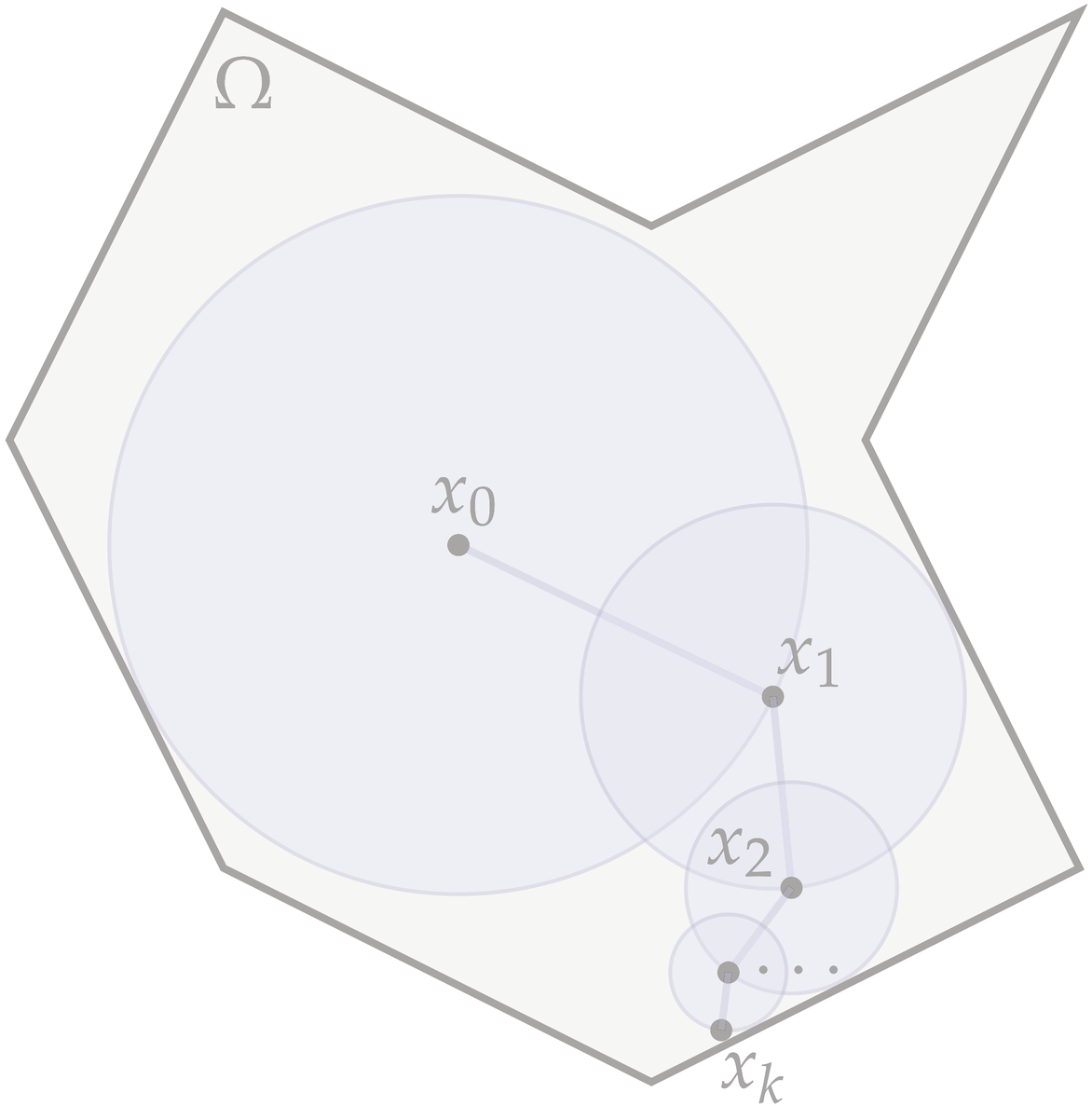
Dirichlet condition



$u = g$ on $\partial\Omega_D$

Prescribe given values

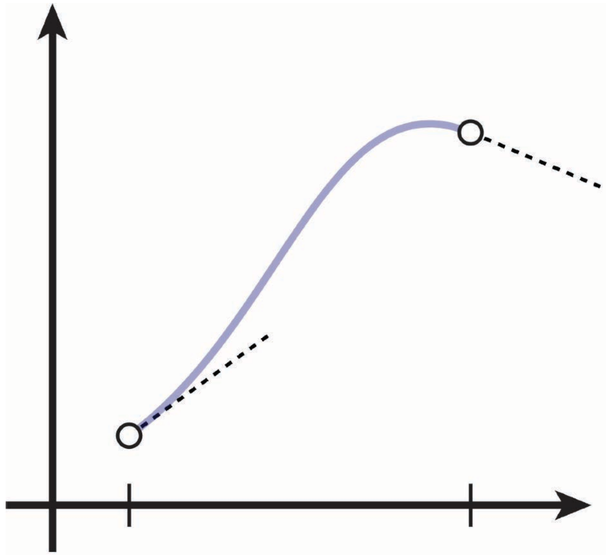
What about Neumann boundary conditions?



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

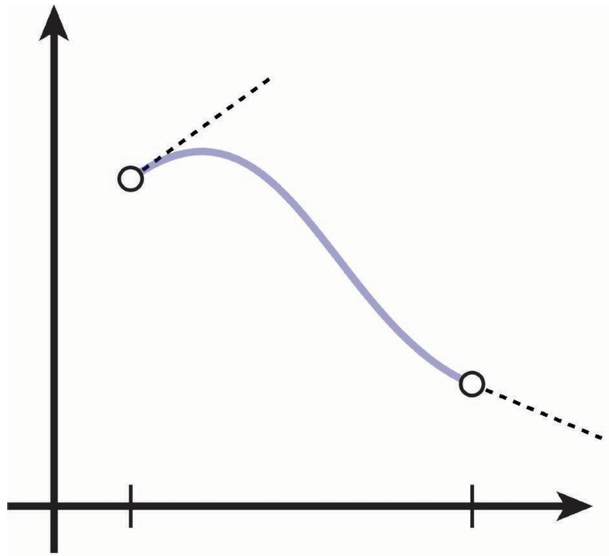
$$u = g \quad \text{on } \partial\Omega_D$$ Dirichlet condition

$$\frac{\partial u}{\partial n} = h \quad \text{on } \partial\Omega_N$$ Neumann condition



$u = g$ on $\partial\Omega_D$

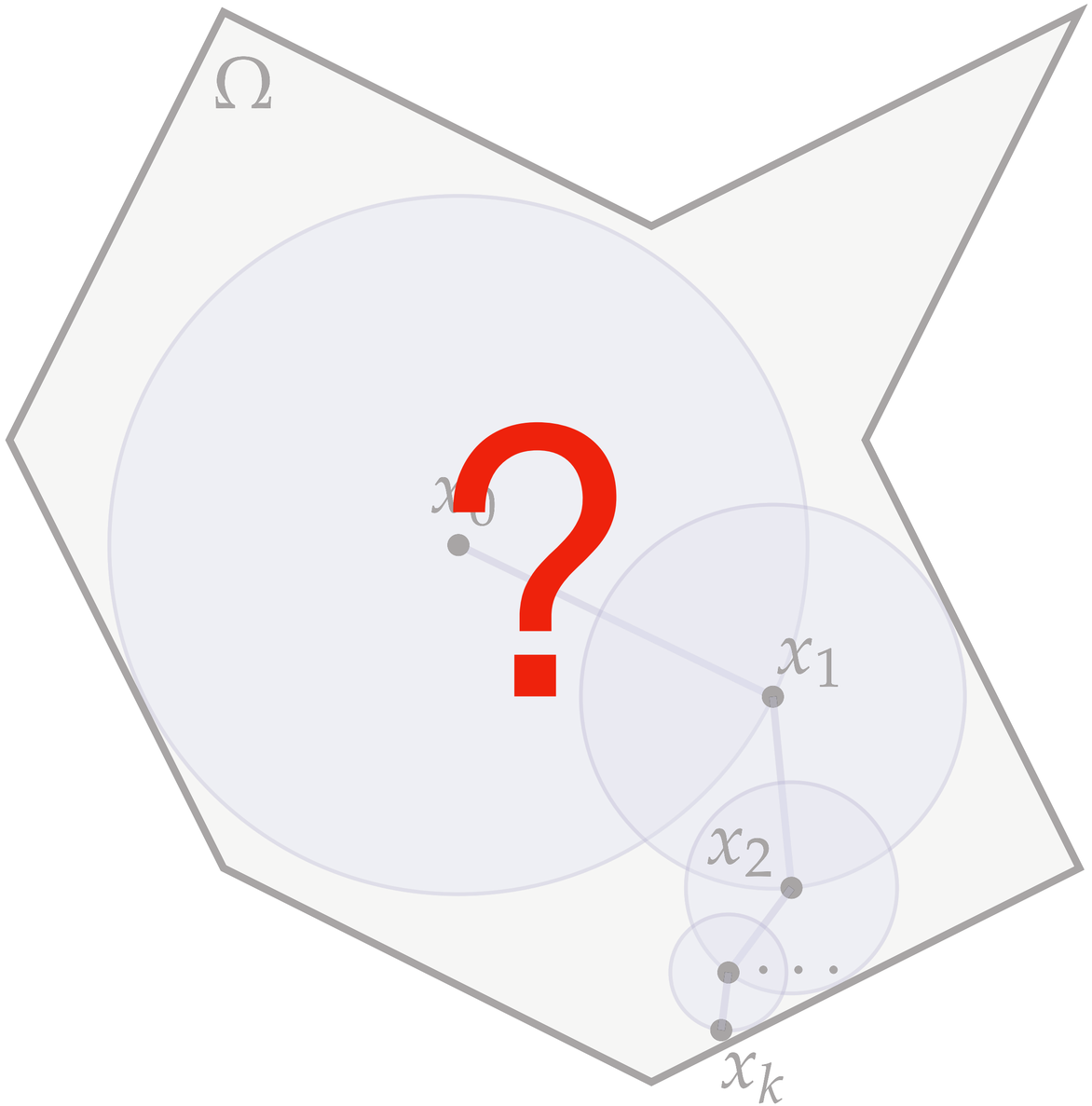
Prescribe given values



$\partial u / \partial n = h$ on $\partial\Omega_N$

Prescribe given derivatives

What about Neumann boundary conditions?



$$\boxed{\begin{aligned} \Delta u &= 0 && \text{on } \Omega \\ u &= g && \text{on } \partial\Omega_D \\ \frac{\partial u}{\partial n} &= h && \text{on } \partial\Omega_N \end{aligned}}$$

Laplace eq.

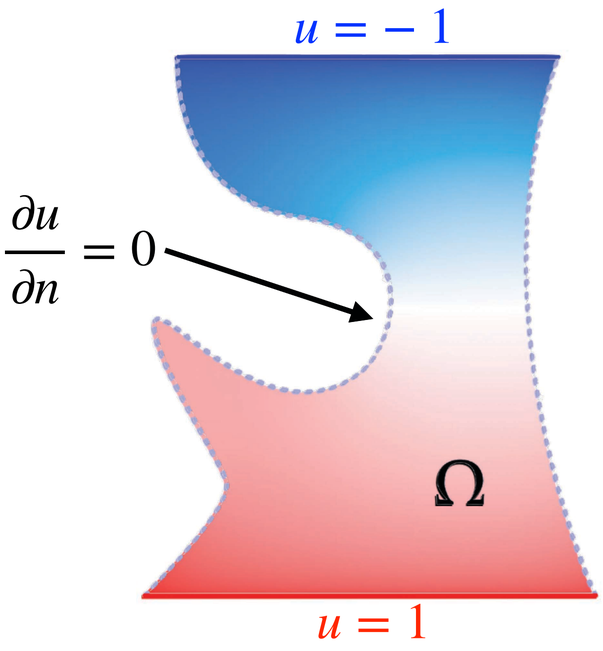
Dirichlet condition

Neumann condition

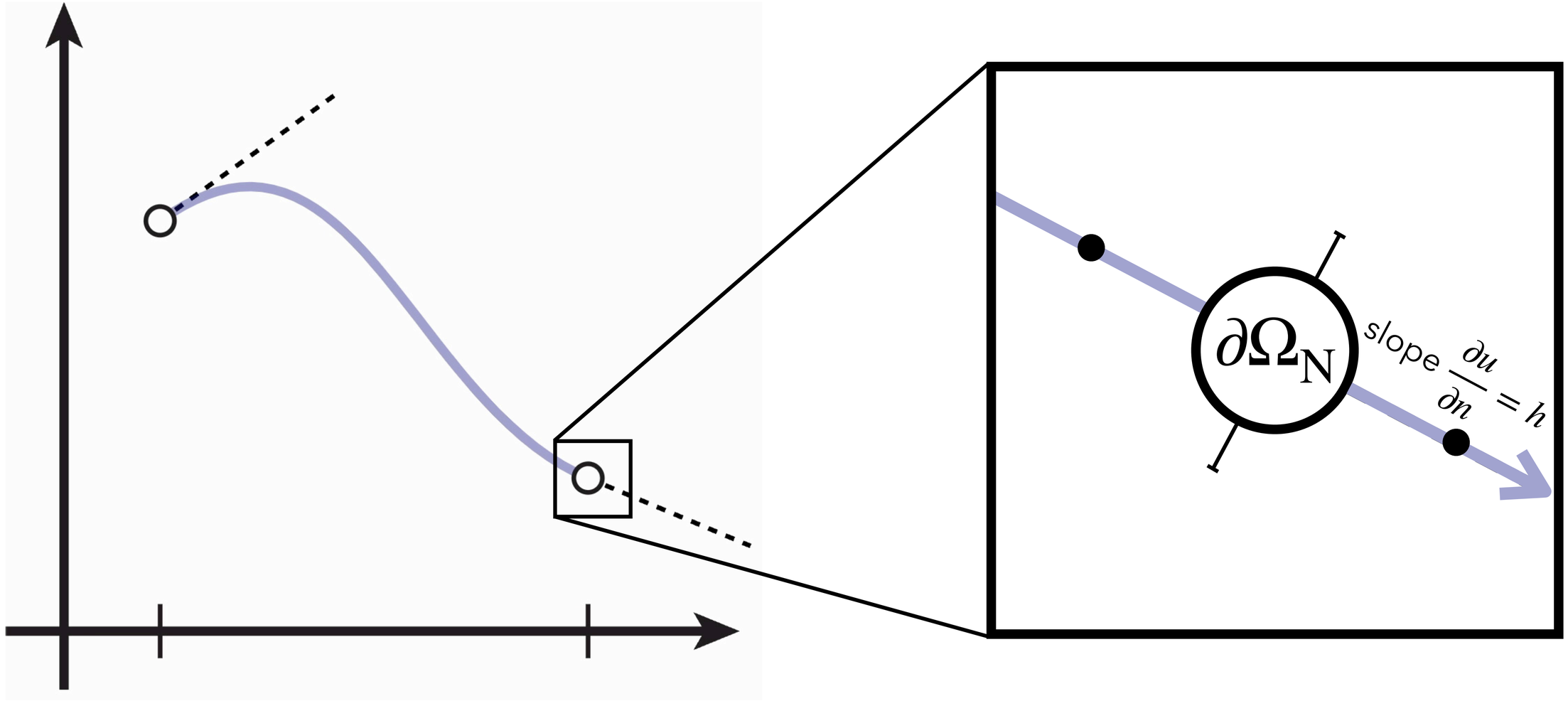
Fluid mechanics:
velocity/pressure gradient

Structural analysis:
surface traction

Thermodynamics:
heat flux



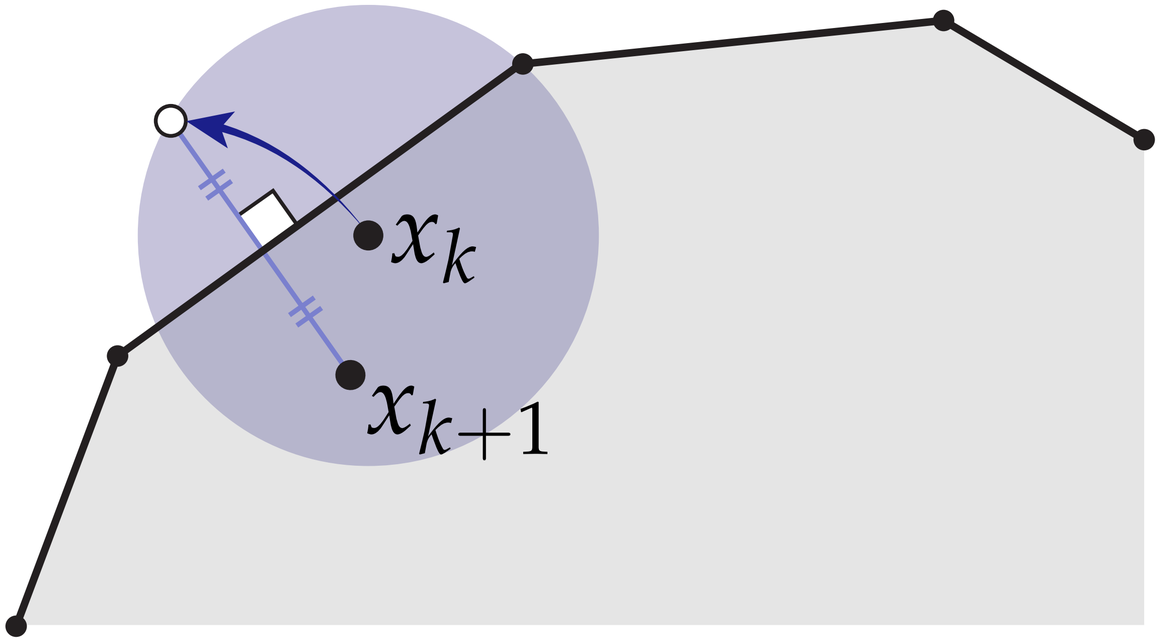
What about Neumann boundary conditions?

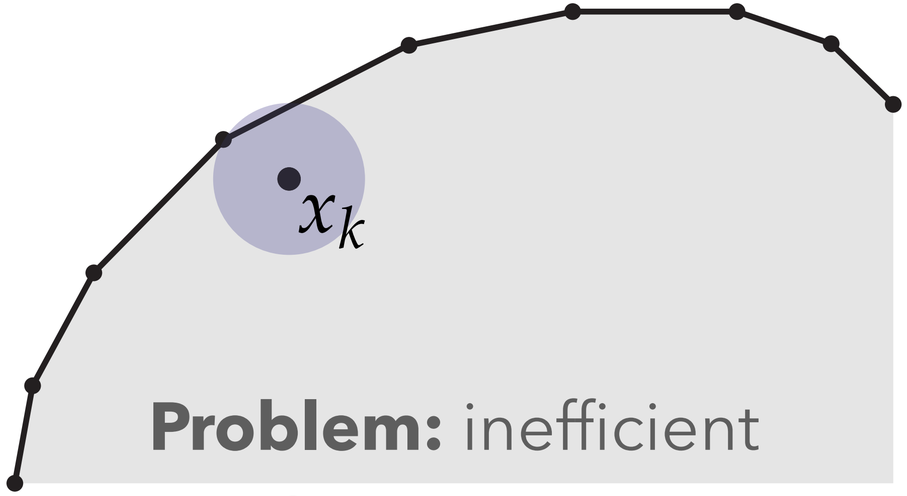


Observation: function extrapolated with slope h will mirror values across $\partial\Omega_{\mathrm{N}}$

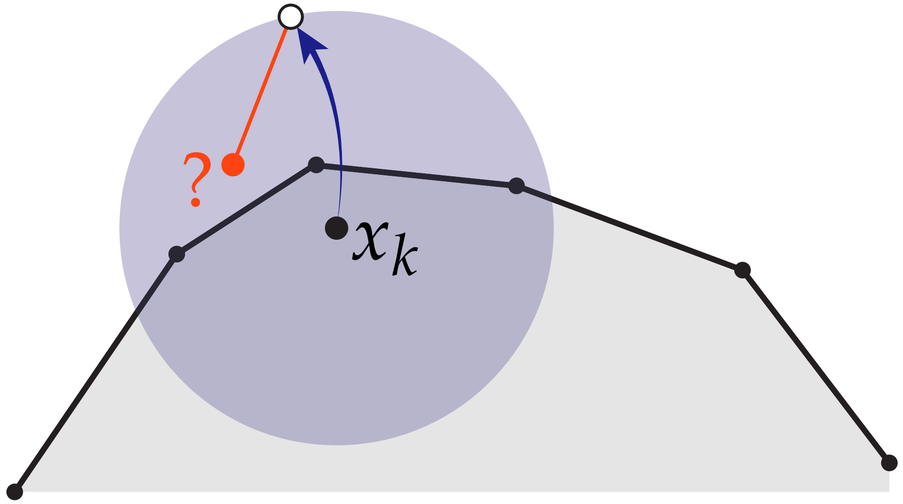
Naïve attempt at reflected random walks

Idea: sample biggest sphere around x crossing one edge & reflect if necessary





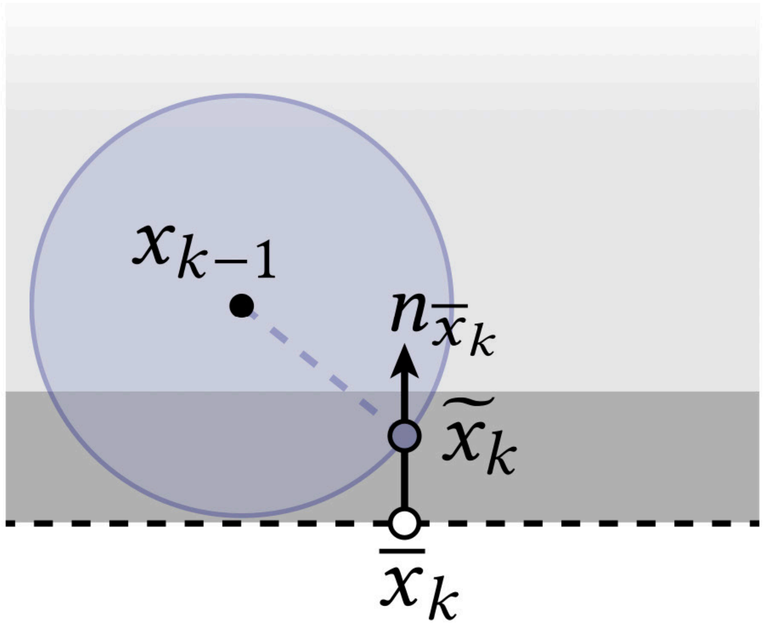
Problem: inefficient



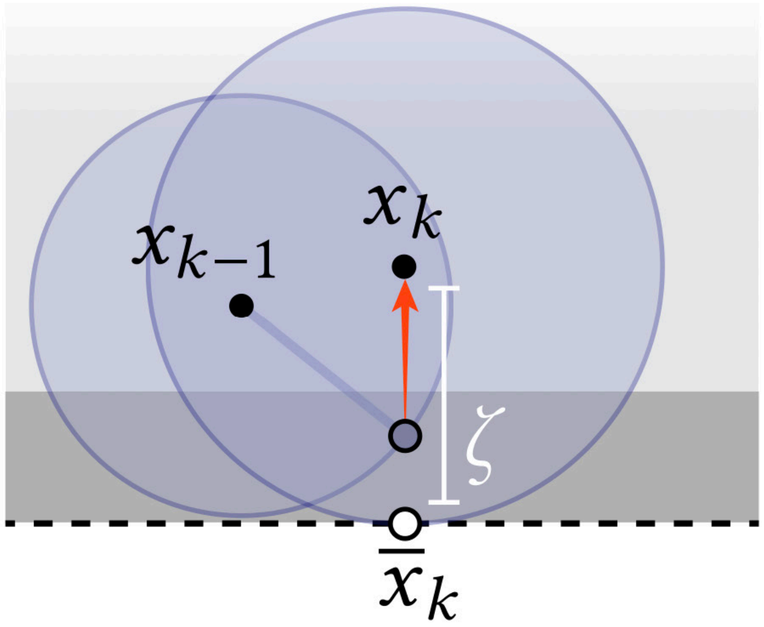
Problem: incorrect/biased

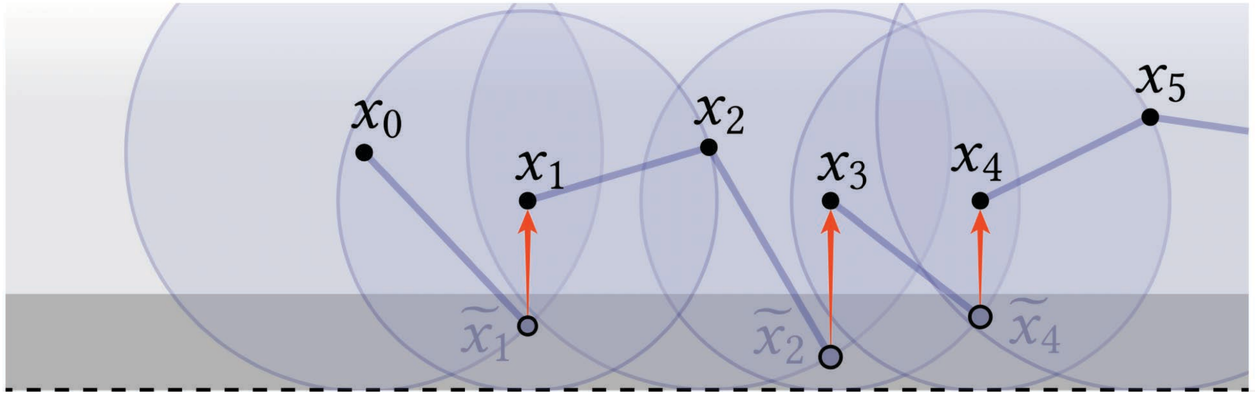
Naïve attempt at reflected random walks

Idea: push walk into the domain by fixed distance [Mascagni & Simonov, 2004]



nudge into domain





$$\frac{du(x)}{dn} \approx \frac{u(x+\zeta n) - u(x)}{\zeta}$$

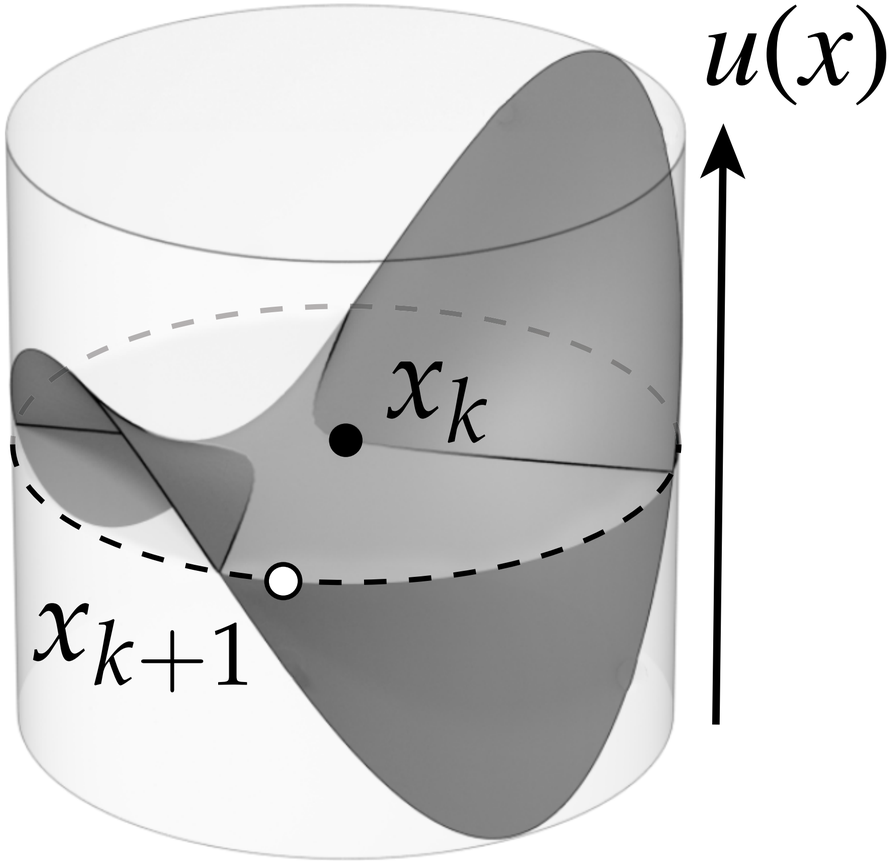
ε-shell

$\partial\Omega_N$

Mean Value Property

A harmonic function u satisfies:

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

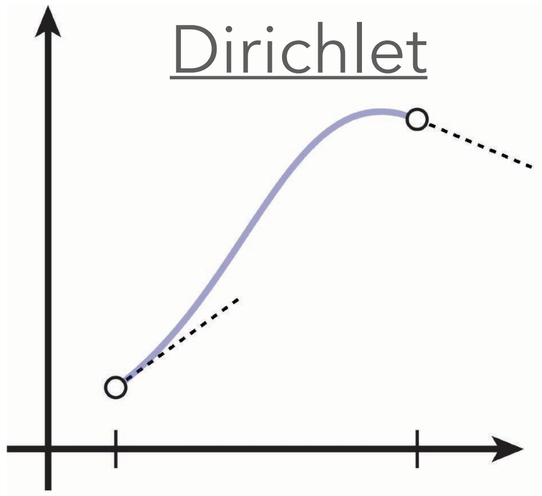


Boundary Integral Equation

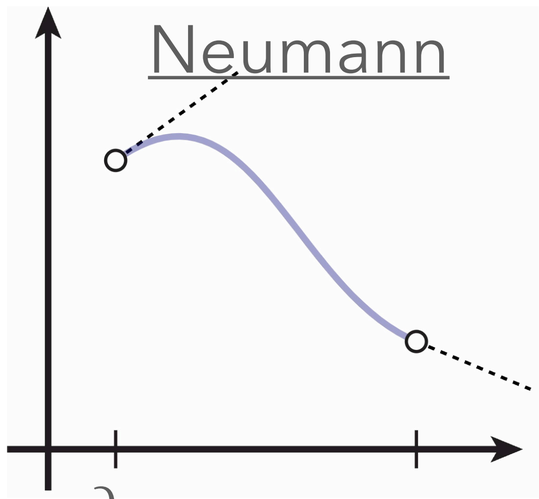
In general, a harmonic function u on domain Ω also satisfies:

$$u(x) = \int_{\partial\Omega} \overbrace{P(x,y)}^{\text{Poisson kernel}}\, u(y) - \overbrace{G(x,y)}^{\text{Greens fn}} \frac{\partial u}{\partial n_y}\, dy$$

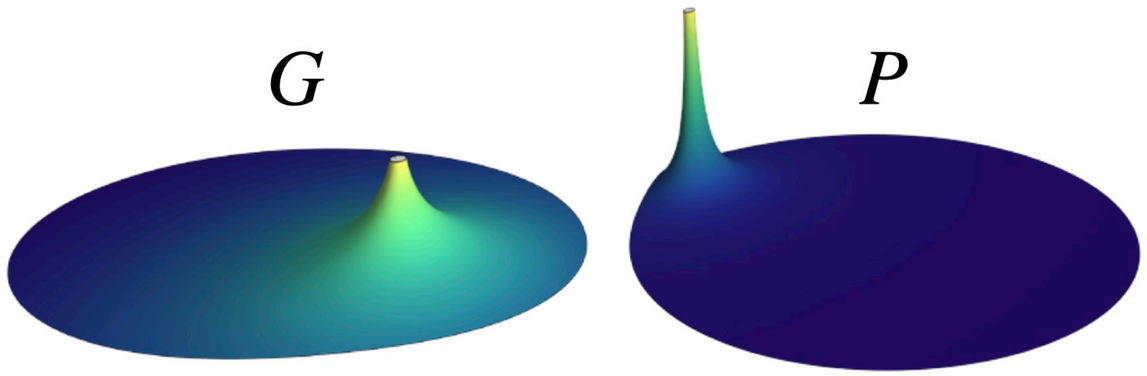
a.k.a. Green's representation theorem



$u = g$ on $\partial\Omega_D$



$\frac{\partial u}{\partial n} = h$ on $\partial\Omega_N$



$$P(x,y) := \frac{\partial G}{\partial n_y}(x,y)$$

Boundary Integral Equation – derivation

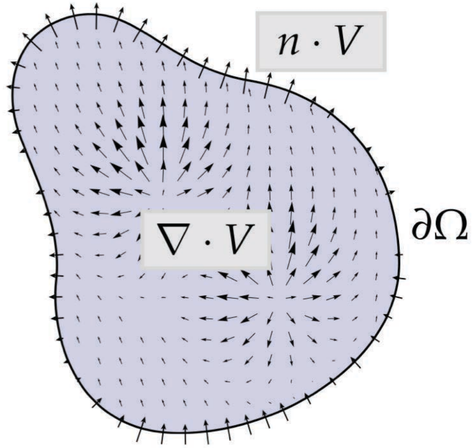
Step 1: Start with Laplace equation $\Delta u = 0$, and multiply both sides by G:

$$\int_\Omega G(x,y)\ \Delta u\ dy = 0$$

Step 2: Apply <u>integration by parts</u>:

$$\int_\Omega \nabla \cdot \big(G(x,y)\ \nabla u\big)\, dy\ -\ \int_\Omega \nabla G(x,y) \cdot \nabla u(y) = 0$$

$$\boxed{\int_\Omega \nabla \cdot V = \int_{\partial\Omega} n \cdot V}$$



Step 3: Apply <u>divergence theorem</u> to the first term:

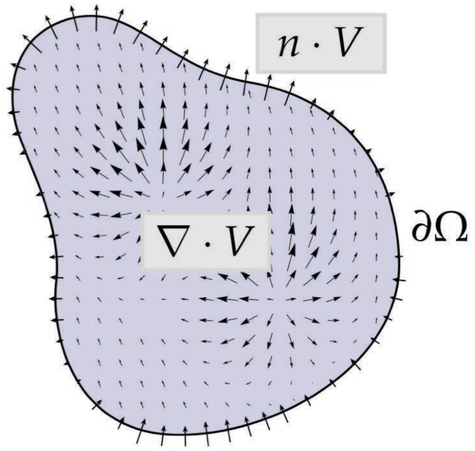
$$\int_{\partial\Omega} G(x,y)\ \frac{\partial u}{\partial n_y}\ dy\ -\ \int_\Omega \nabla G(x,y) \cdot \nabla u(y) = 0$$

Boundary Integral Equation – derivation

Step 3: Apply <u>divergence theorem</u> to the first term:

$$\int_{\partial\Omega} G(x,y)\,\frac{\partial u}{\partial n_y}\,dy \;-\; \int_{\Omega} \nabla G(x,y)\cdot\nabla u(y) = 0$$

$$\boxed{\int_{\Omega} \nabla\cdot V = \int_{\partial\Omega} n\cdot V}$$



Step 4: Apply <u>integration by parts</u> again to the second term

$$\int_{\Omega} \Delta G(x,y)\;u(y)\;dy = \int_{\partial\Omega} P(x,y)\;u(y) \;-\; G(x,y)\,\frac{\partial u}{\partial n_y}\,dy$$

Step 6: By definition, $\Delta G(x,y) = \delta_x(y)$ and we get

$$u(x) = \int_{\partial\Omega} P(x,y)\;u(y) \;-\; G(x,y)\,\frac{\partial u}{\partial n_y}\,dy$$

Boundary Integral Equation

For a Poisson equation $\Delta u = f$, we can express solution as:

Poisson kernel

$$u(x) = \int_{\partial\Omega} P(x,y)\, u(y)\, dy \quad \text{Dirichlet values}$$

$$- \int_{\partial\Omega} G(x,y) \frac{\partial u}{\partial n_y}\, dy \quad \text{Neumann values}$$

Greens fn

$$+ \int_{\Omega} G(x,y)\, f(y)\, dy \quad \text{source term}$$

Can we now apply Monte Carlo directly to this BIE?

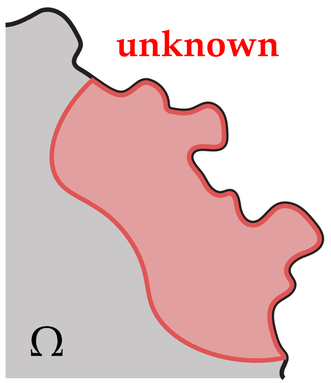
Problems:

– u not known on $\partial\Omega_{\mathrm{N}}$

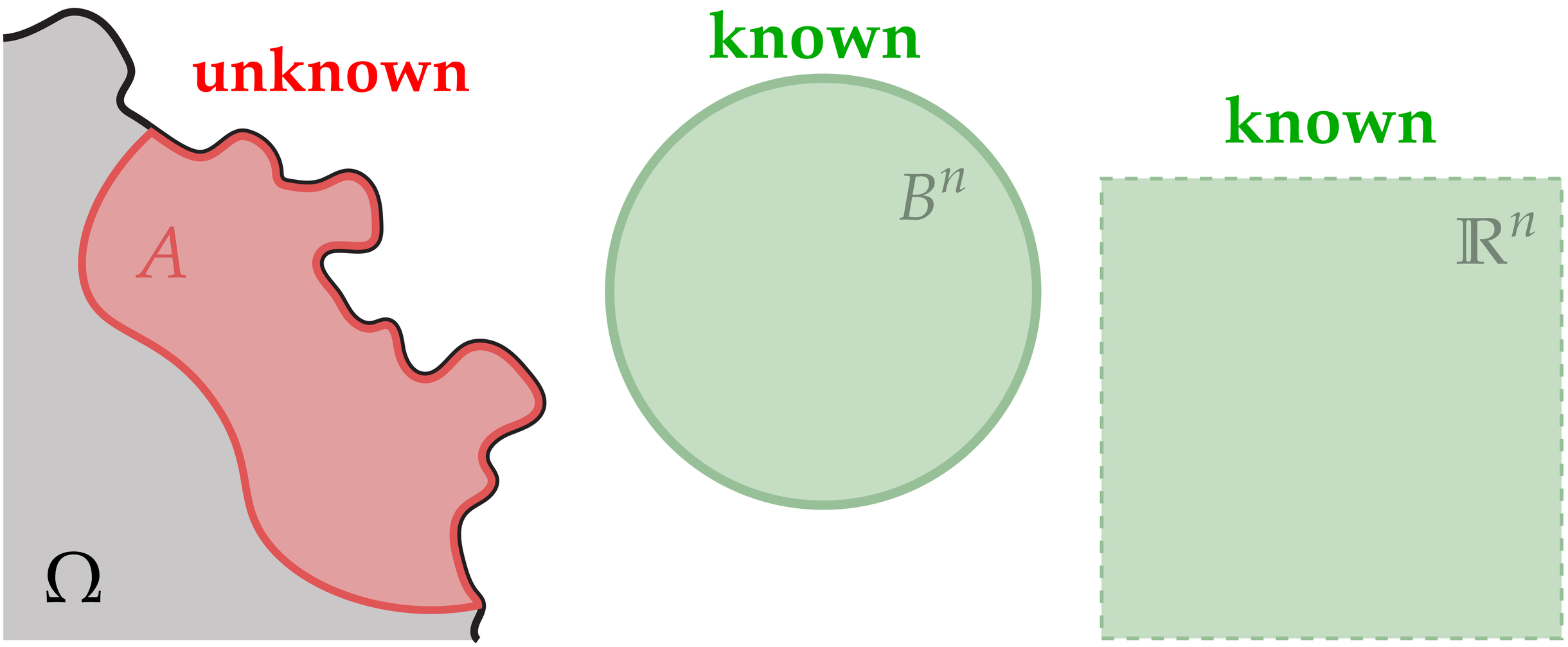
– $\frac{\partial u}{\partial n}$ not known on $\partial\Omega_{\mathrm{D}}$

– do not have random walk procedure

– P & G **never** known for arbitrary Ω



Poisson kernel & Green's function



Mean value property is a special case of BIE

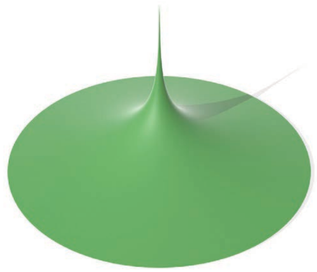
For a Poisson equation $\Delta u = f$, we can express solution as:

$$u(x) = \int_{\partial\Omega} P(x,y)\, u(y)\, dy$$

$$- \int_{\partial\Omega} G(x,y) \frac{\partial u}{\partial n_y}\, dy$$

$$+ \int_{\Omega} G(x,y)\, f(y)\, dy$$

Let $\Omega = B \implies P(x,y) = \dfrac{1}{|\partial B|}$



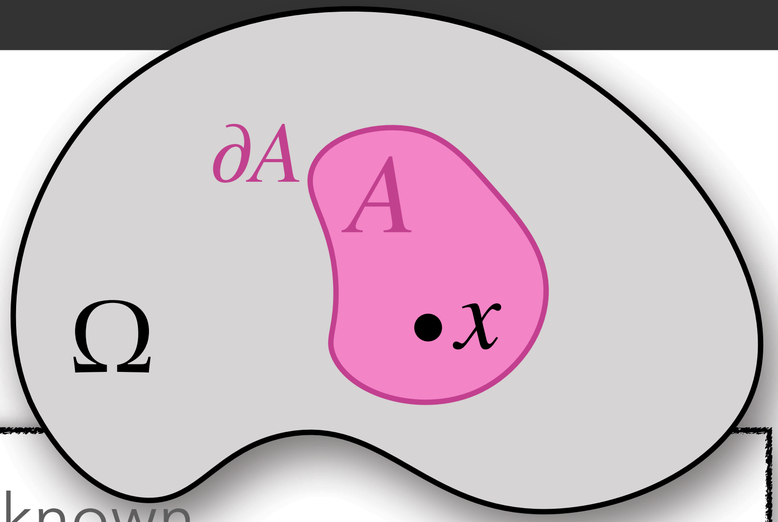
$\implies G(x,y) = 0 \quad \text{on } \partial B$

$\implies$ **mean value property**

Boundary Integral Equation

Laplace equation (zero Neumann)

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



Poisson kernel of A

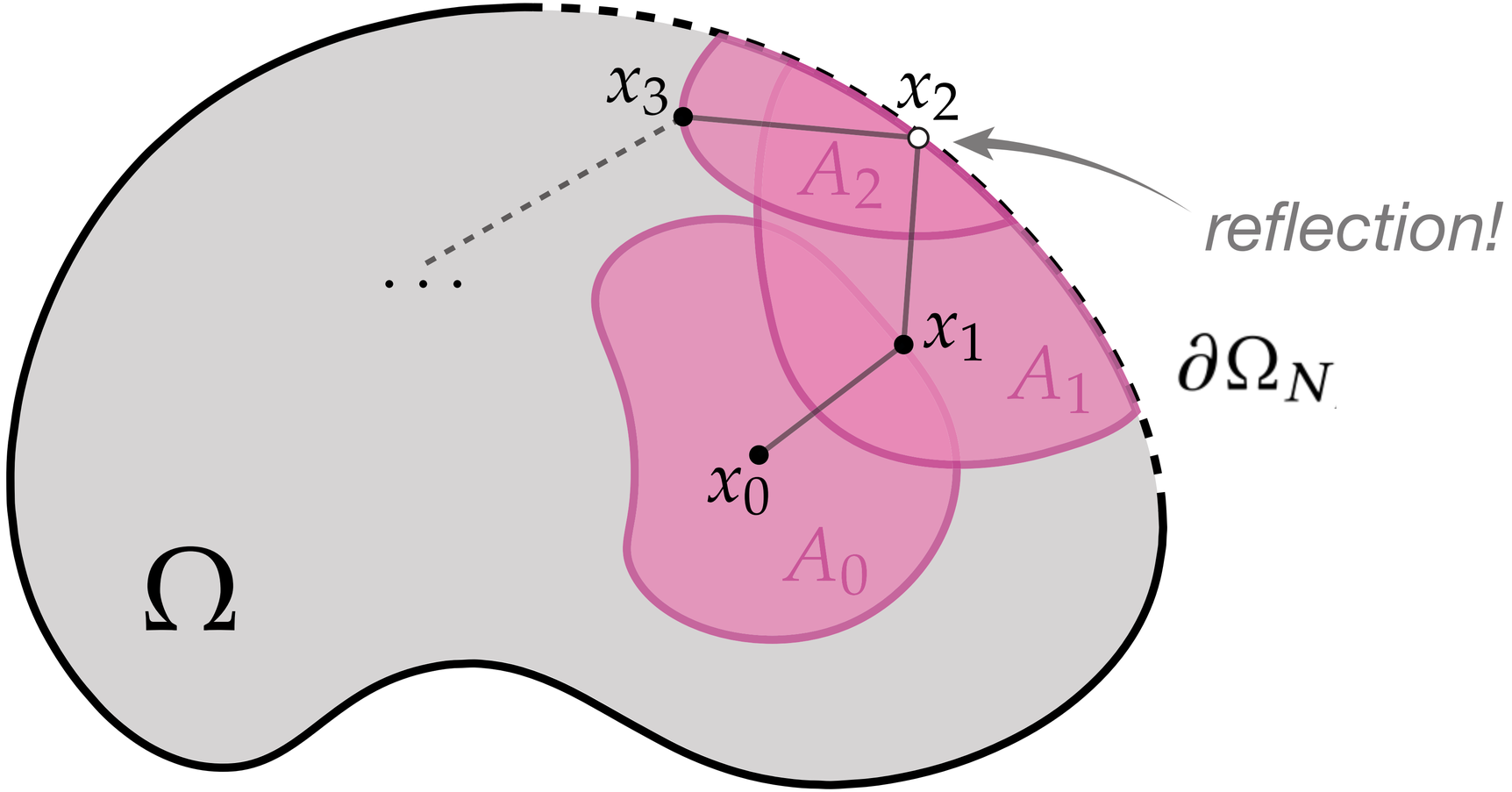
unknown value at y

$$u(x) = \int_{\partial A} P(x, y)\, u(y)\, dy$$

Intuition: "generalized mean value property"

Walk on subdomains

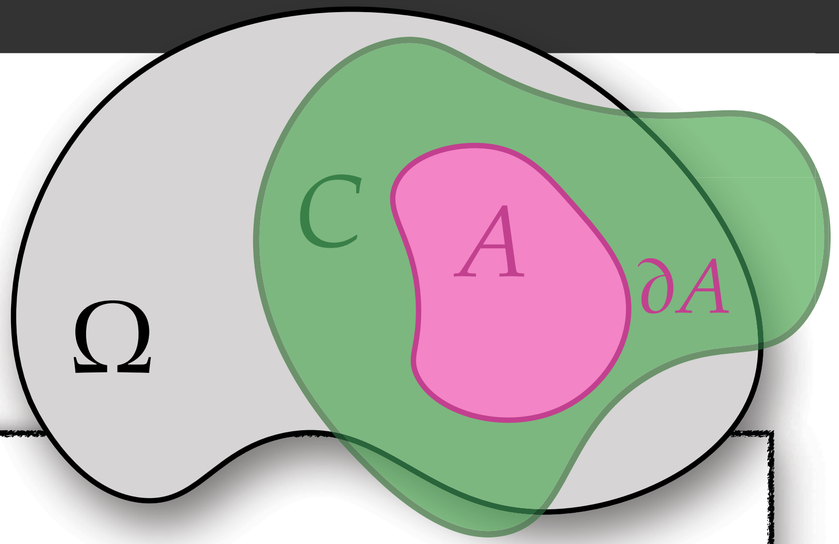
Next idea: take a "walk" on subdomains that contain the Neumann boundary (by sampling Poisson kernel)



Boundary Integral Equation (General)

Laplace equation (zero Neumann)

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

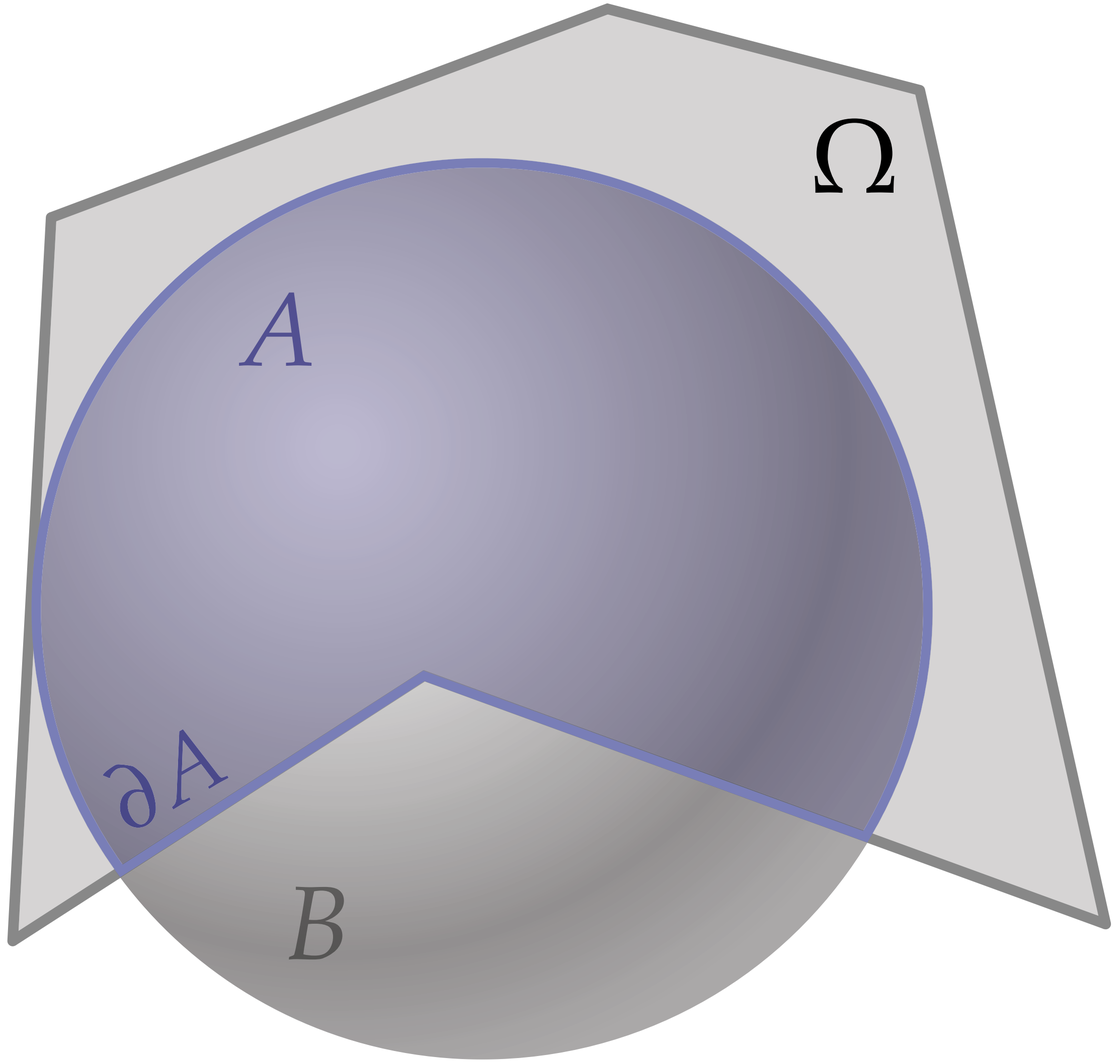


Poisson kernel of C (not A)

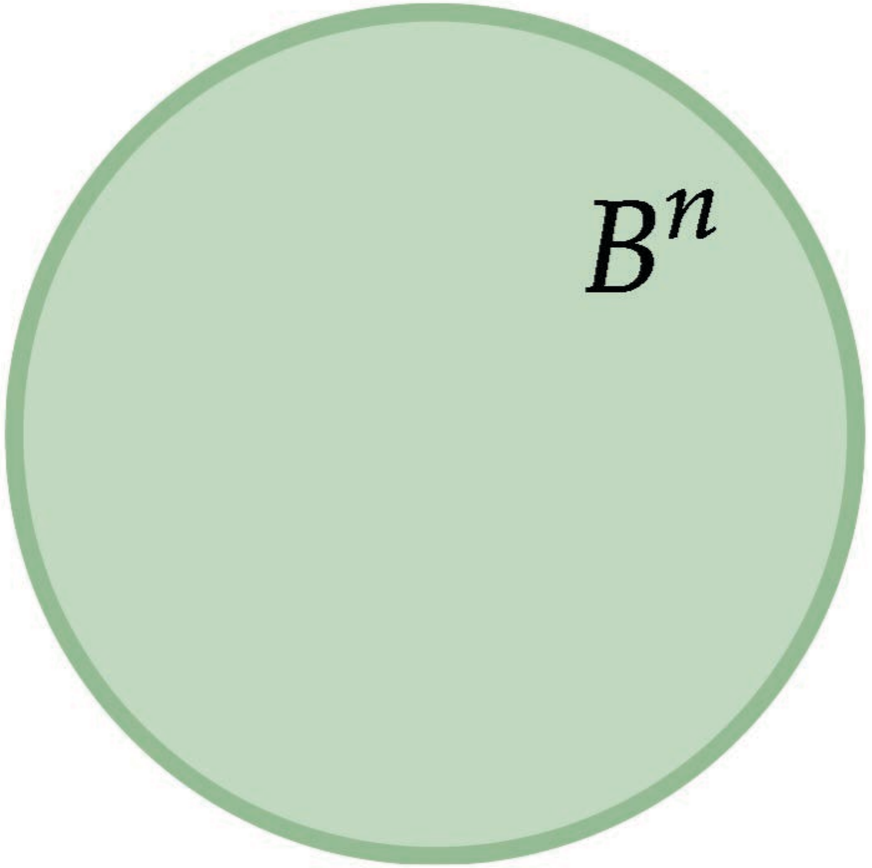
$$u(x) = \int_{\partial A} P^C(x, y)\, u(y)\, dy$$

integrate over boundary of A (not C)

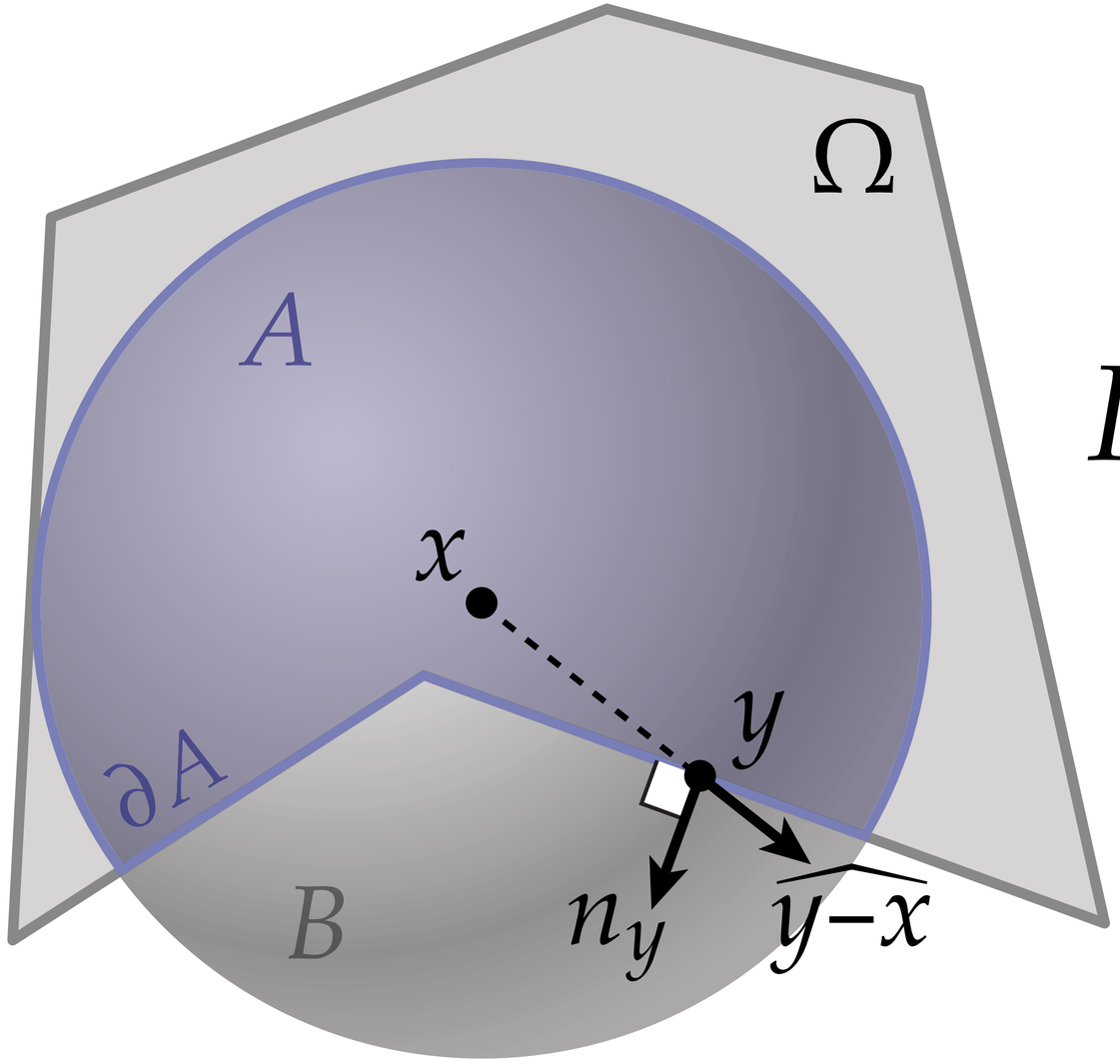
Choice of region C



$C =$ ball



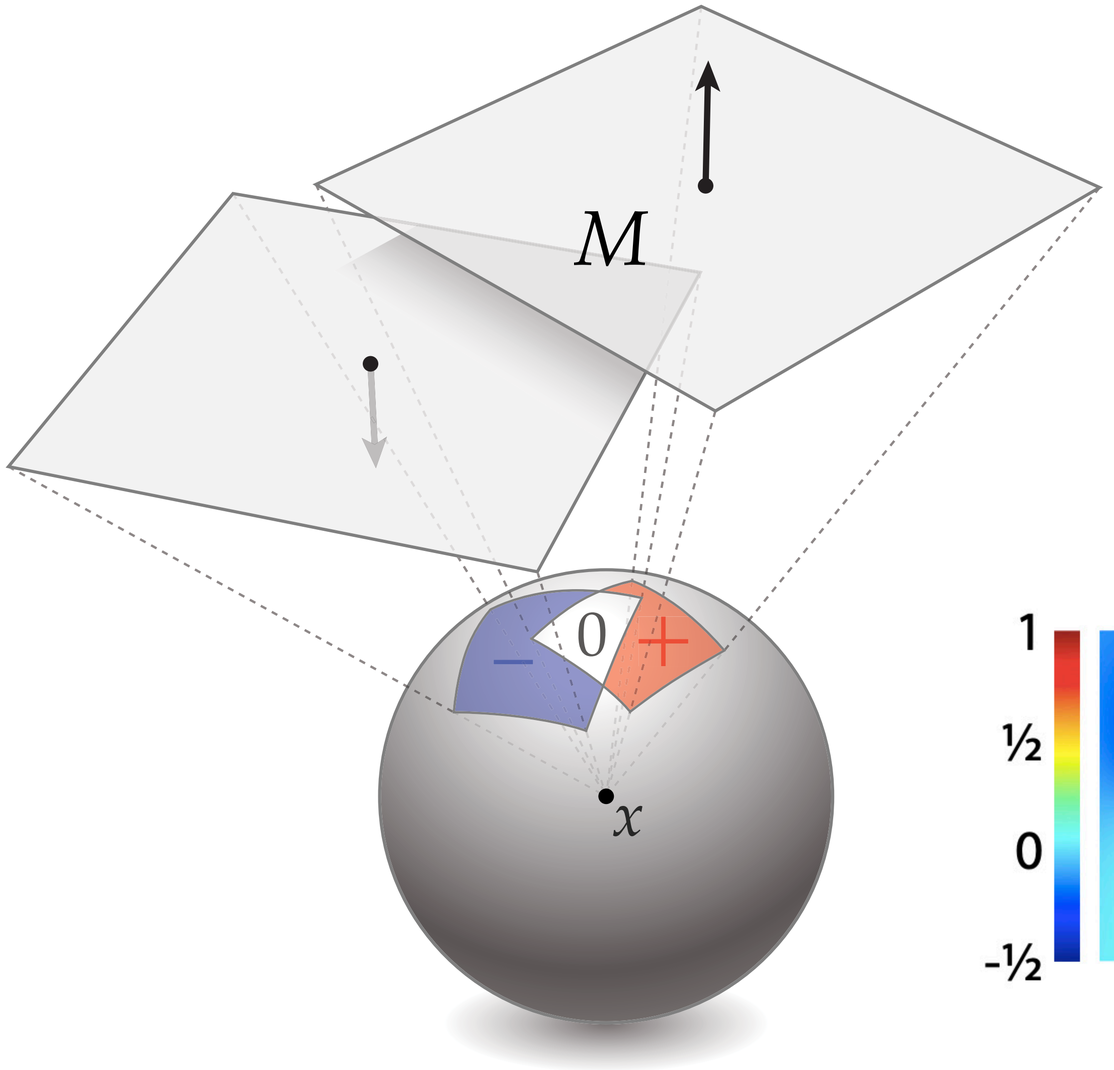
Poisson kernel for a ball



$$P^B|_{\partial A} = \frac{n_y \cdot \widehat{y-x}}{4\pi r^2}$$

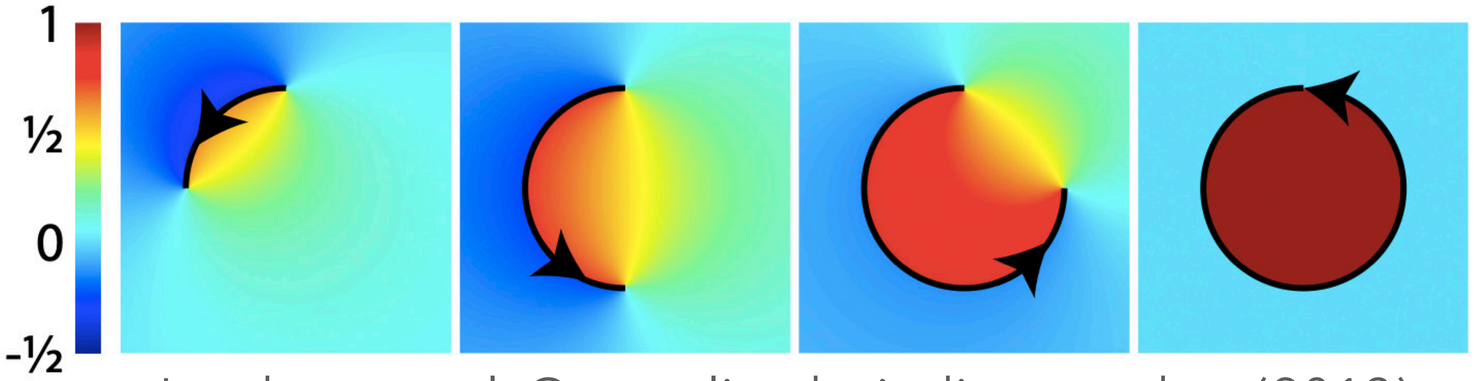
signed solid angle

Signed solid angle



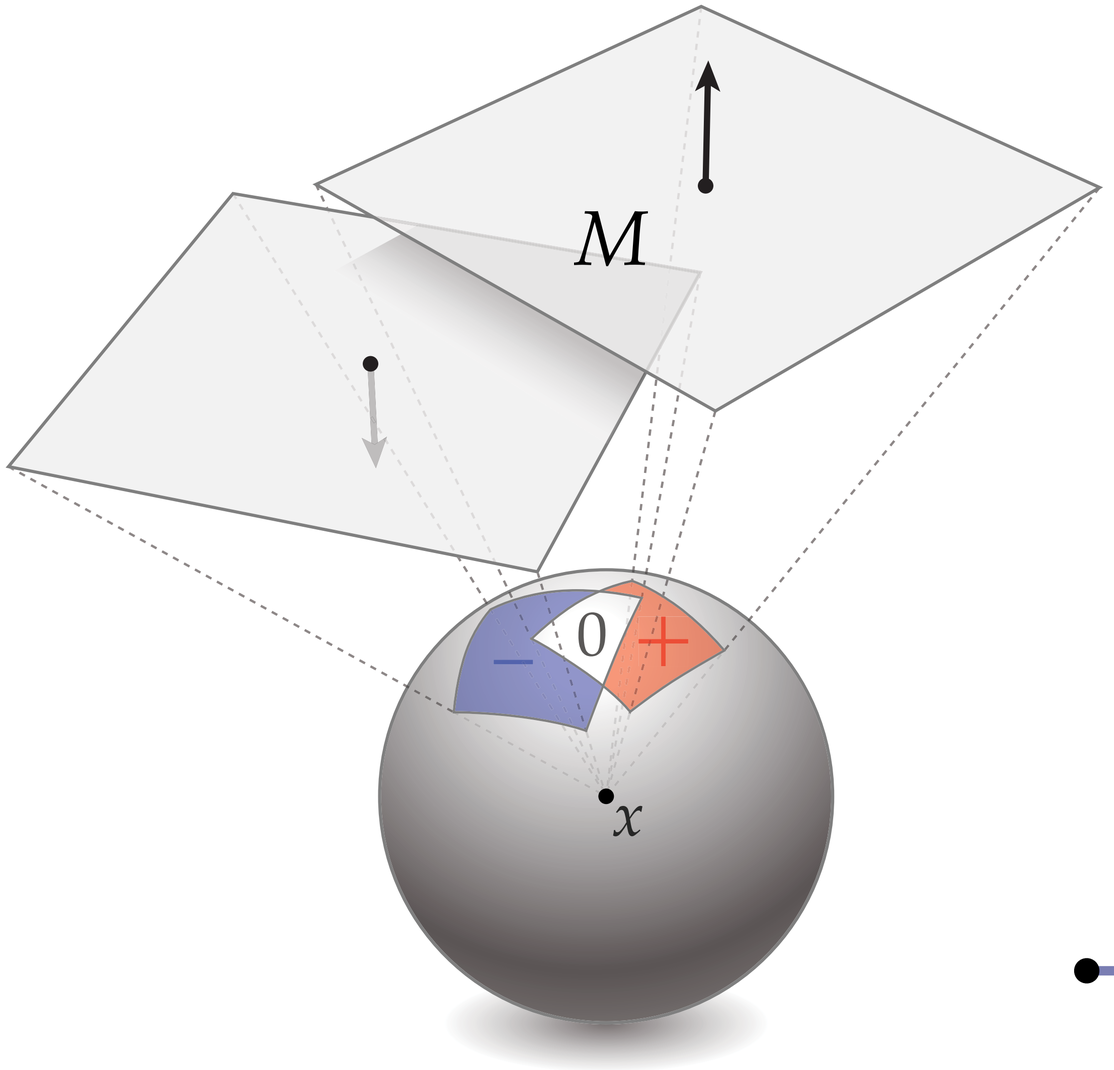
Poisson kernel/solid angle is the kernel of the winding number integral:

$$\int_M \frac{n_y \cdot \widehat{y - x}}{4\pi r^2} \, dA_M$$



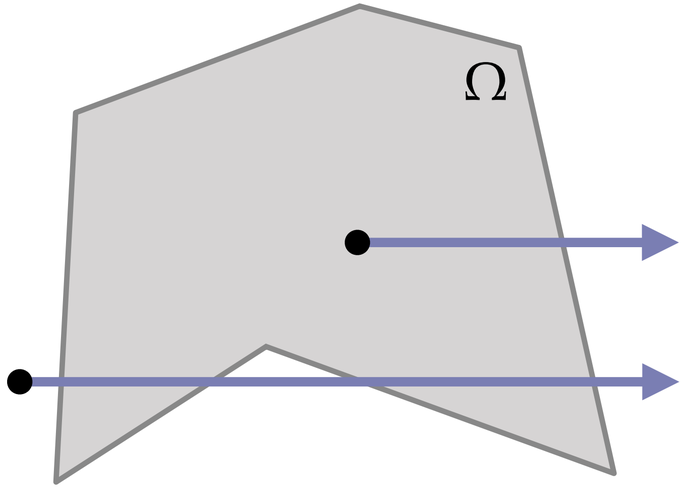
Jacobson et al, Generalized winding number (2013)

Signed solid angle



Poisson kernel/solid angle is the kernel of the winding number integral:

$$\int_M \frac{n_y \cdot \widehat{y - x}}{4\pi r^2} \, dA_M$$



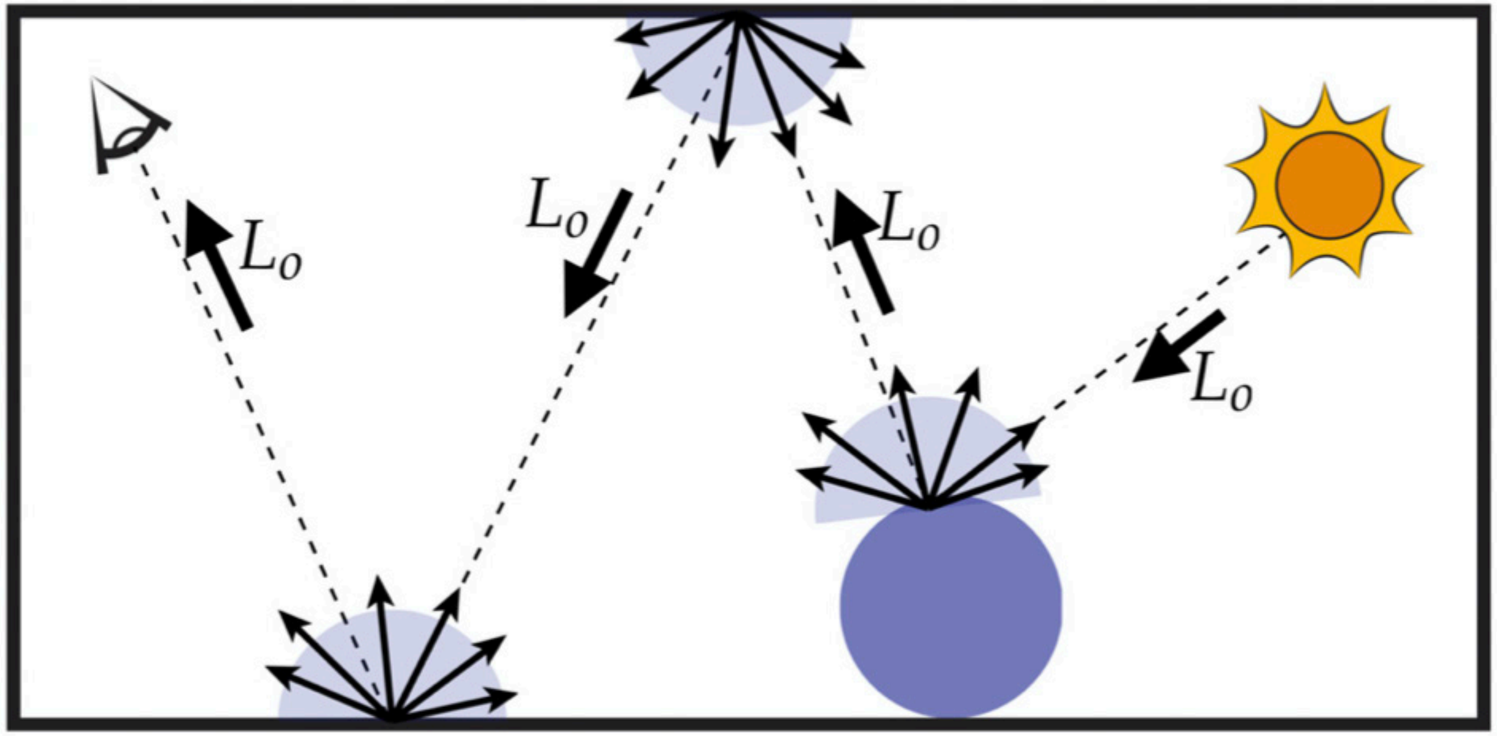
odd #hits $\Longrightarrow$ inside

even #hits $\Longrightarrow$ outside

Sampling signed solid angle in photorealistic rendering

$$L_o(x) = \int_\Omega \underbrace{\rho(x,y)}_{\text{brdf}} \underbrace{L_i(y)}_{\text{radiance}} \underbrace{V(x,y)}_{\text{visibility}} \underbrace{\frac{n_y \cdot \widehat{y-x}}{4\pi r^2}}_{\text{geometry}} dA$$

sample y by tracing ray & returning first (visible) hit!



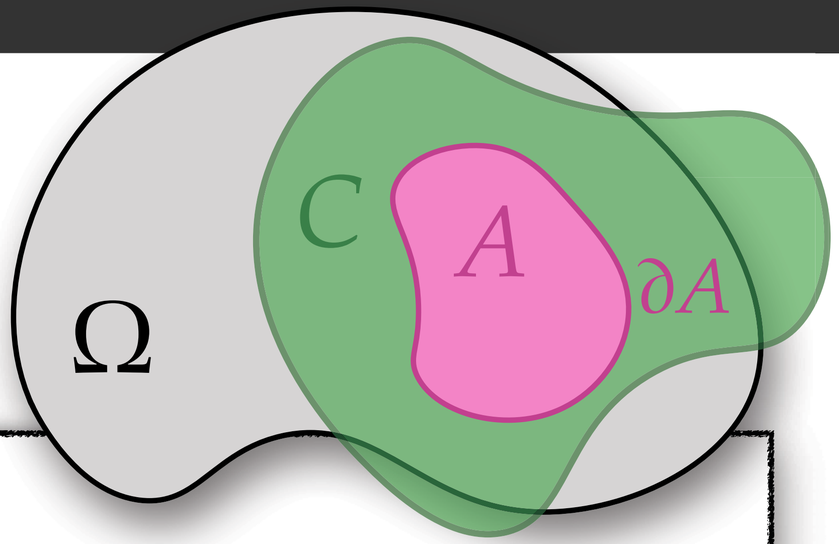
BIE has no visibility term V!

No problem, just **shoot a ray** in a random direction to sample from solid angle

Boundary Integral Equation (General)

Laplace equation (zero Neumann)

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



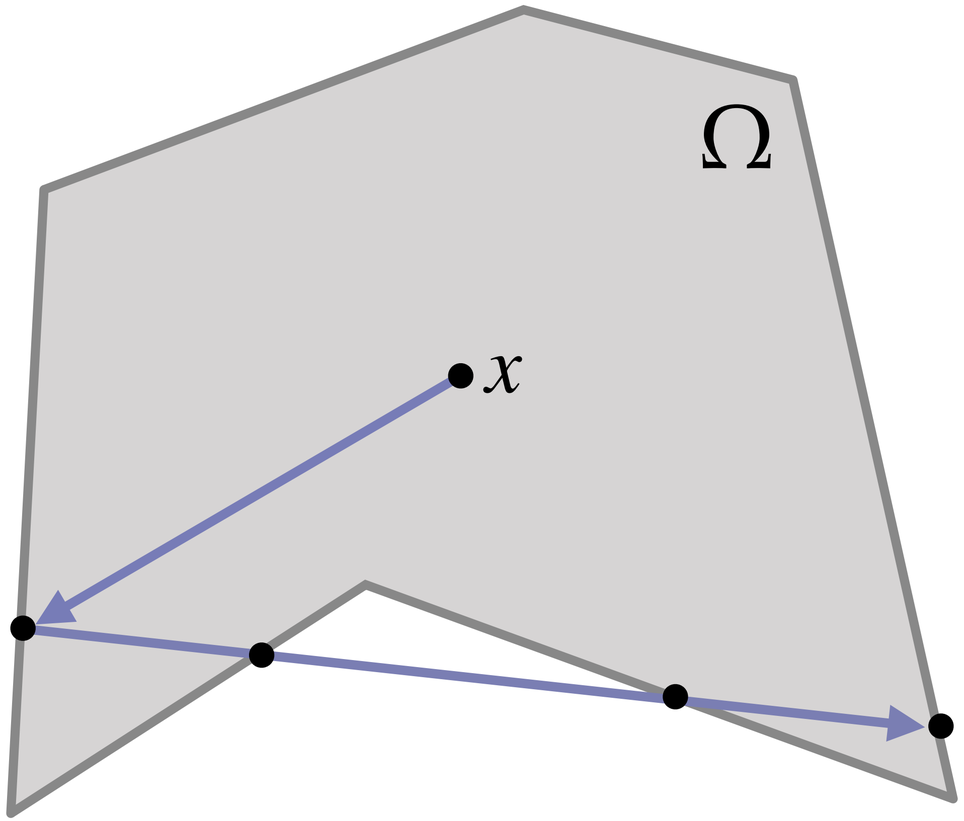
Poisson kernel of C (not A)

$$u(x) = \int_{\partial A} P^C(x, y)\, u(y)\, dy$$

integrate over boundary of A (not C)

Choice of region A

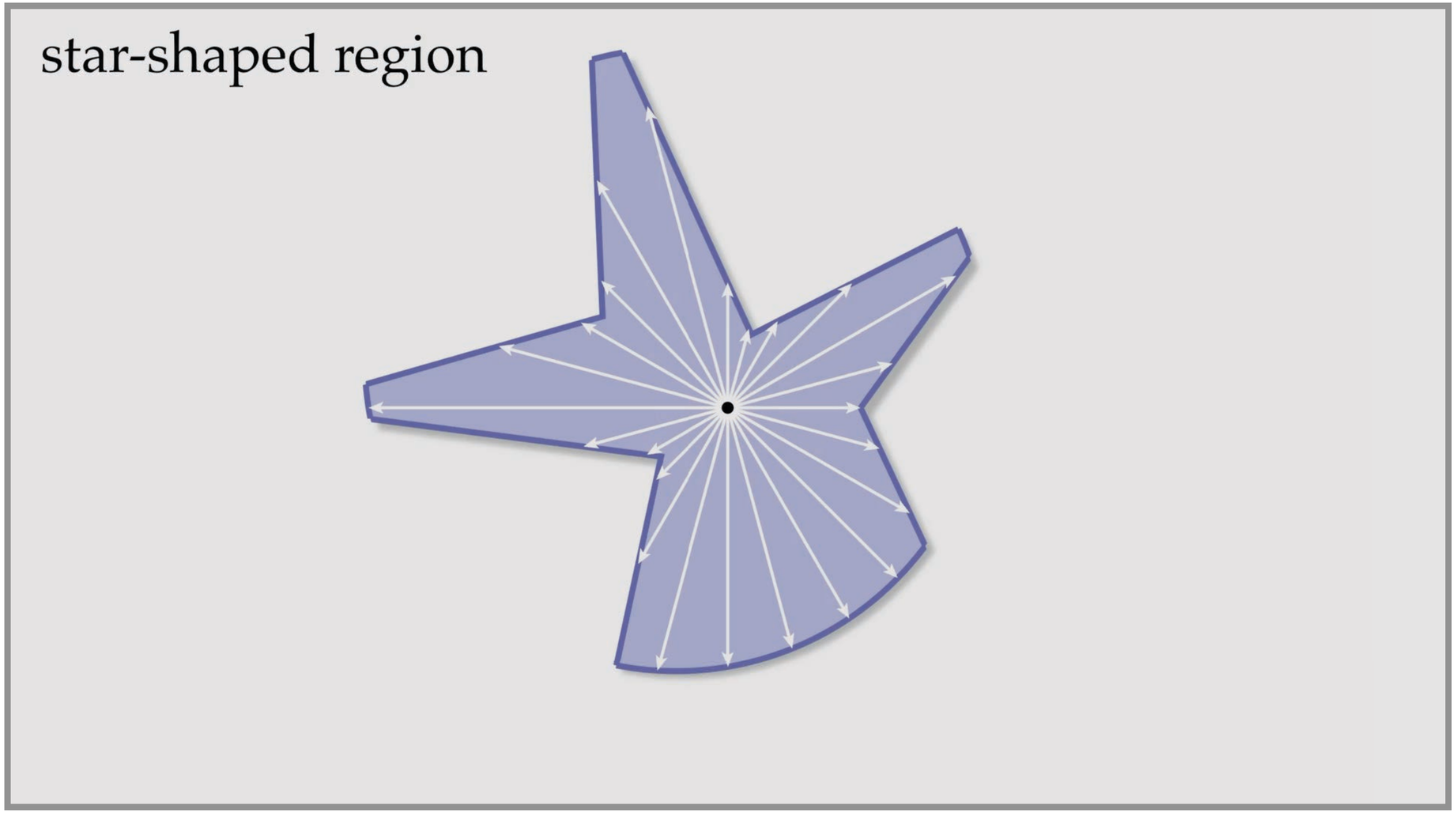
A = **input domain**



Multiple intersections $\Longrightarrow$ ***exponential* growth in points to track**

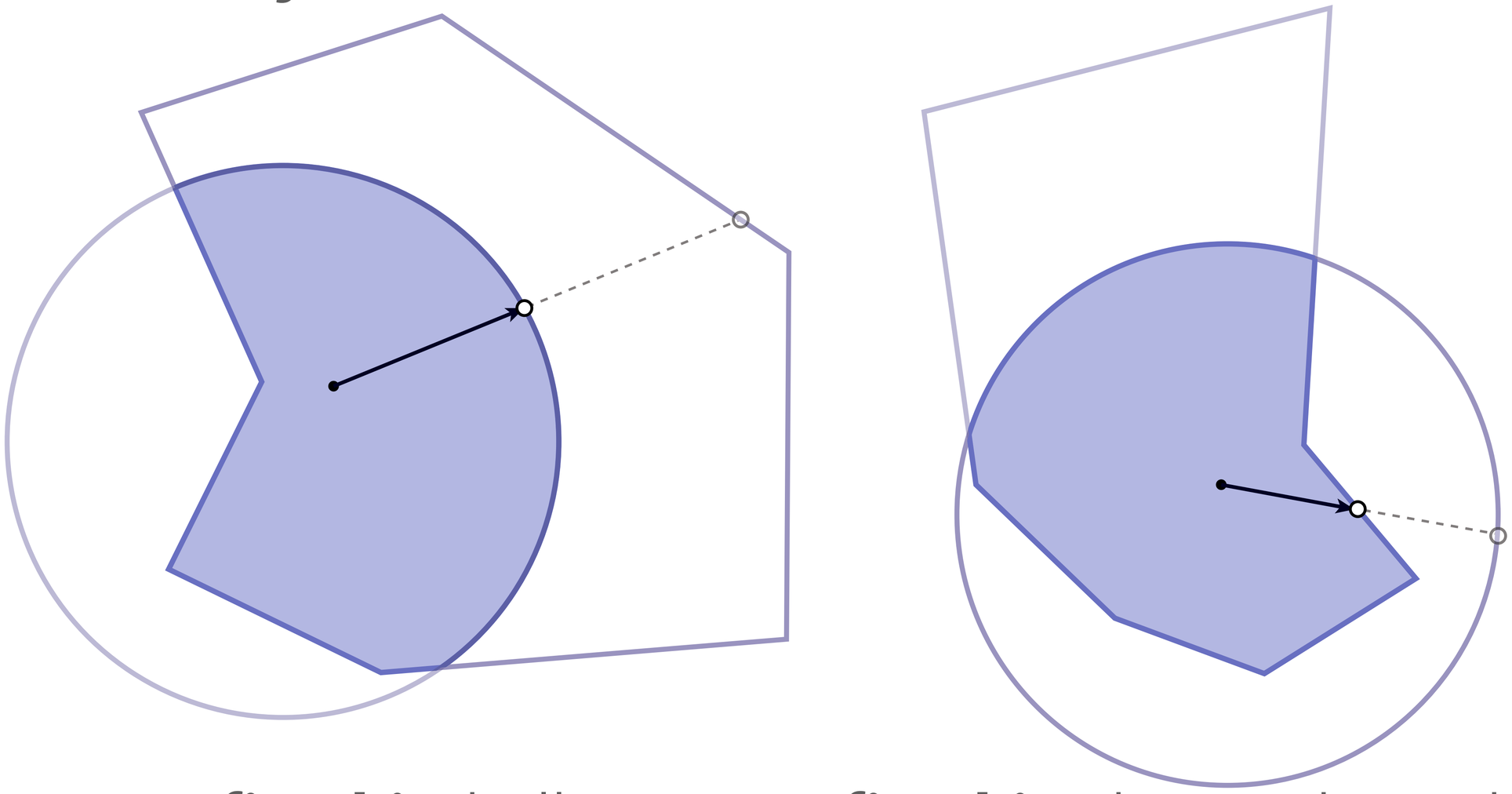
Star-shaped region

A = **star-shaped region,** C = **ball**



Sampling signed solid angle in BIE

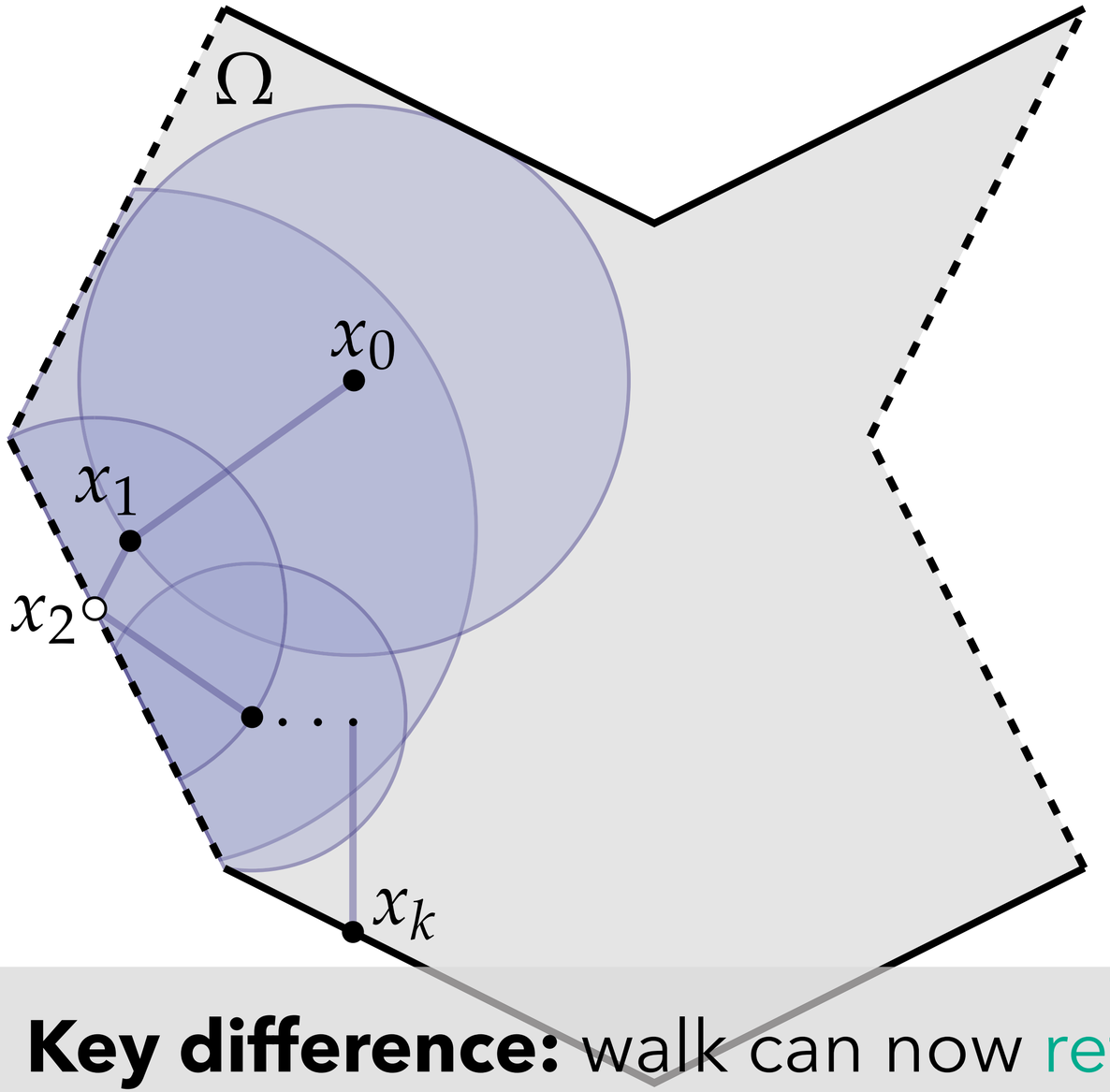
Just **shoot a ray** in a random direction



first hit: ball

first hit: domain boundary

Walk on stars [Sawhney*, Miller*, Gkioulekas†, Crane†, 2023]



$$\boxed{\begin{aligned} \Delta u &= 0 && \text{on } \Omega \\ u &= g && \text{on } \partial\Omega_D \\ \frac{\partial u}{\partial n} &= 0 && \text{on } \partial\Omega_N \end{aligned}}$$

Laplace eq.

Dirichlet ——

Neumann - - - -

until we reach Dirichlet boundary $\partial\Omega_D$:

- find star-shaped region St around x_k
- sample x_{k+1} from ∂St

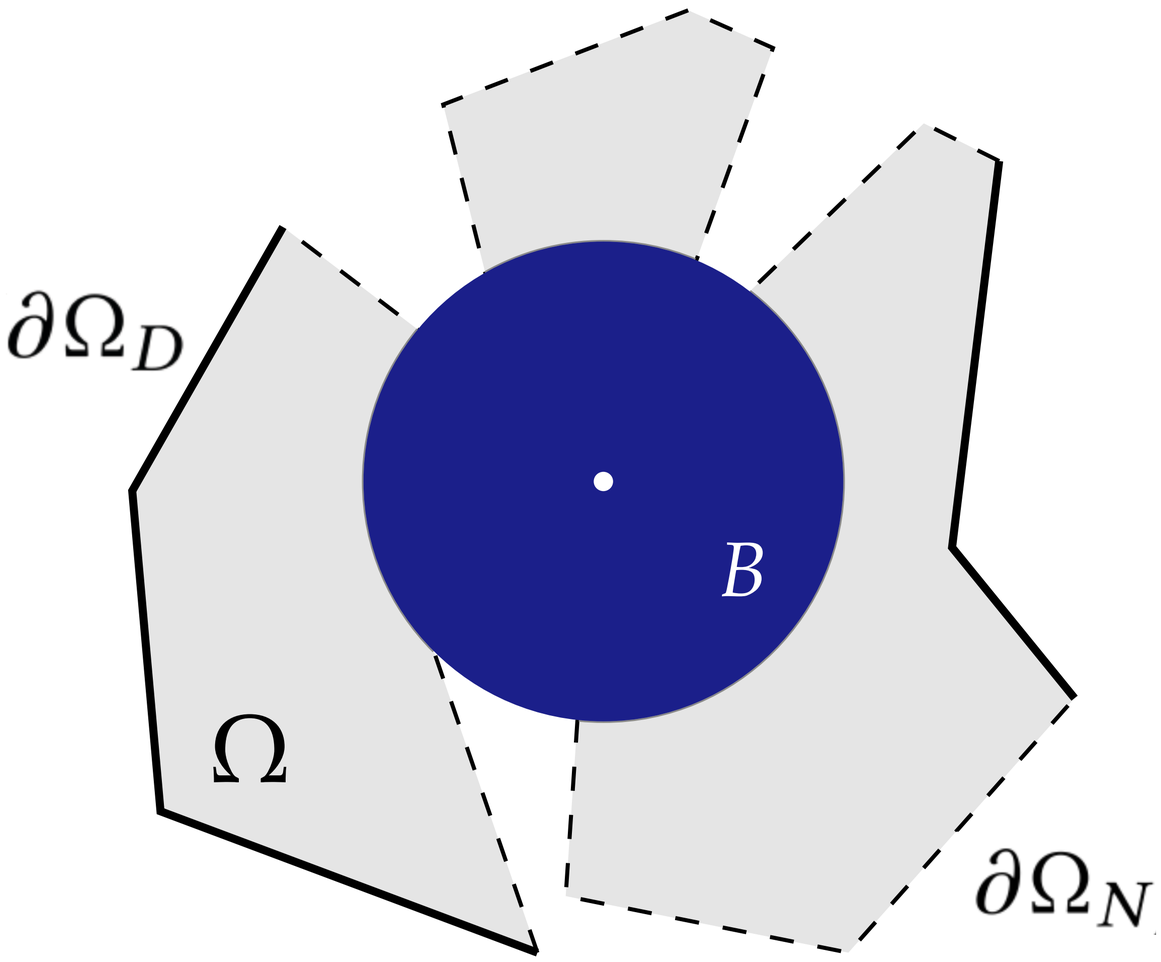
add boundary value $g(x_k)$ to average

(repeat *N* times)

Key difference: walk can now reflect off Neumann boundary

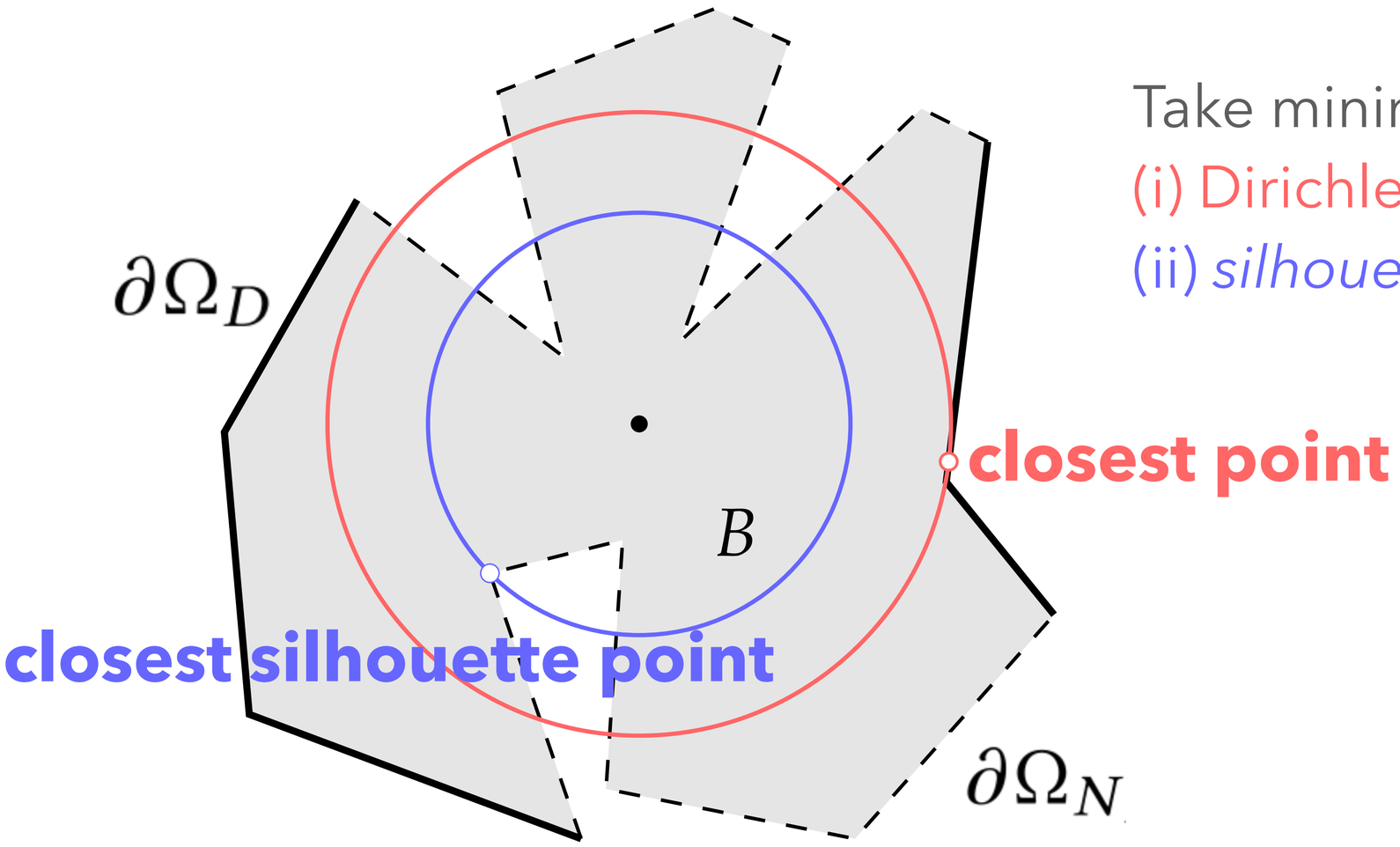
Finding star-shaped regions

How do we find big star-shaped regions?



Finding star-shaped regions

How do we find big star-shaped regions?

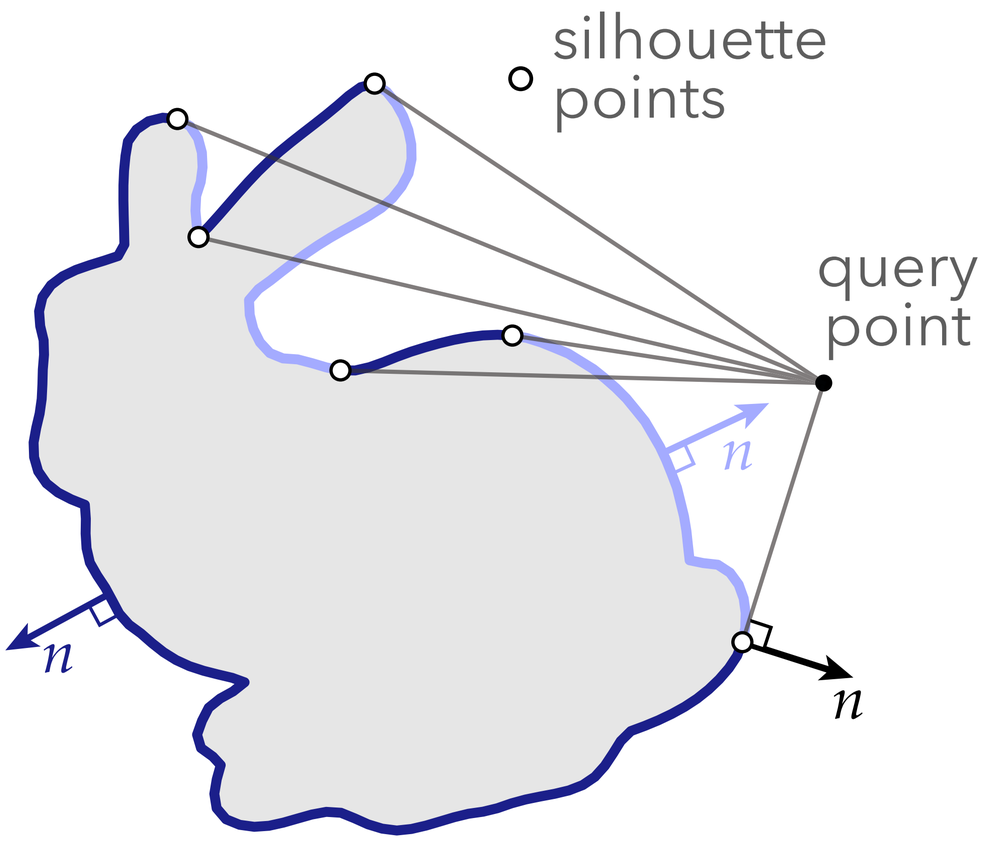


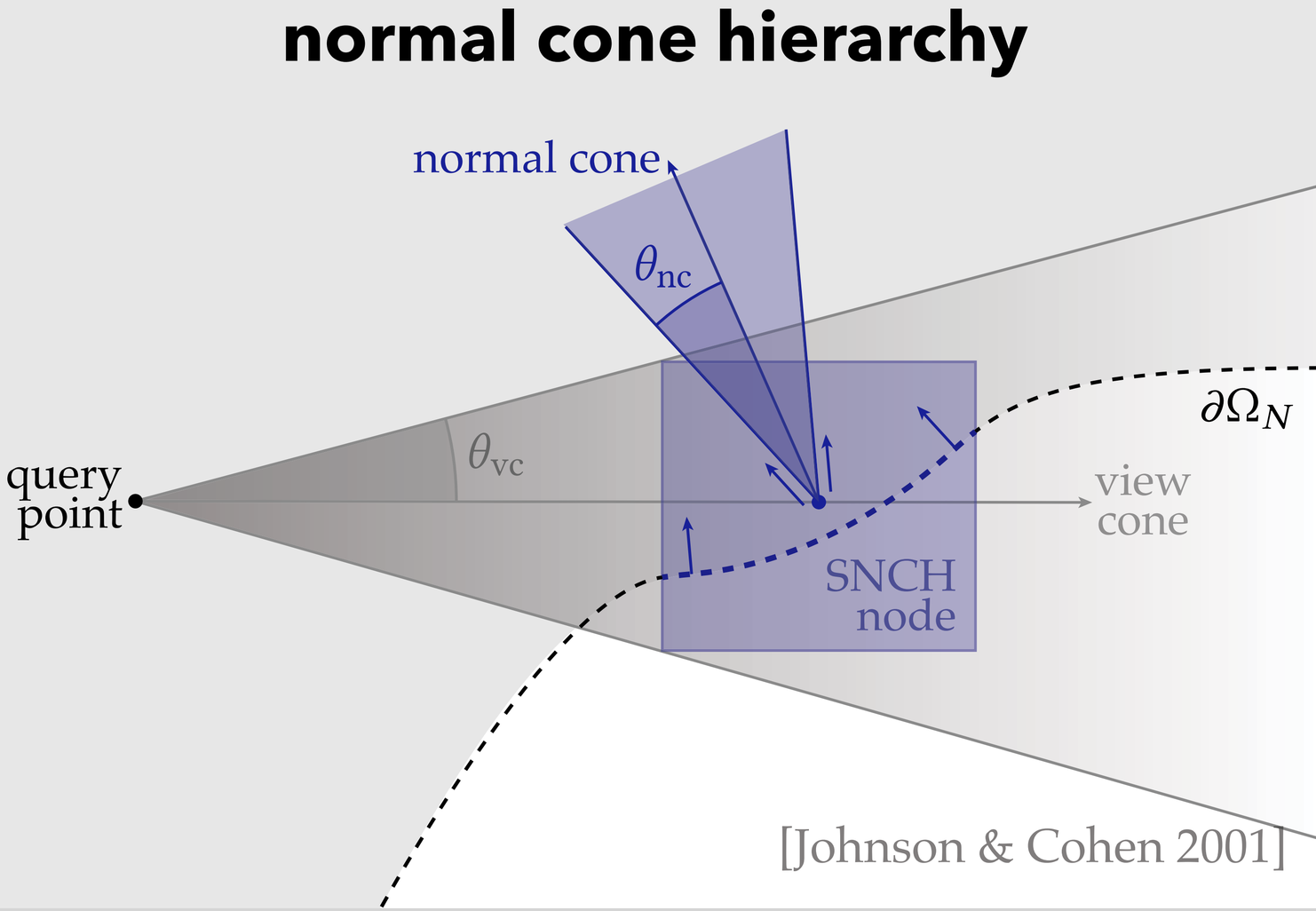
Take minimum distance to:

(i) Dirichlet boundary

(ii) *silhouette* of Neumann boundary

Closest silhouette point queries

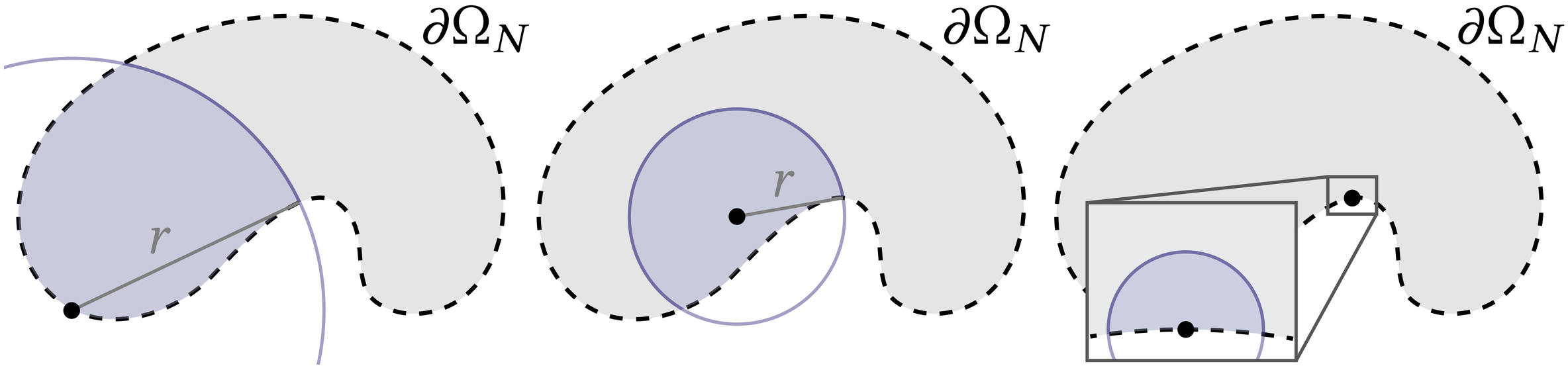




Can re-use same BVH built for ray intersections and closest point queries.

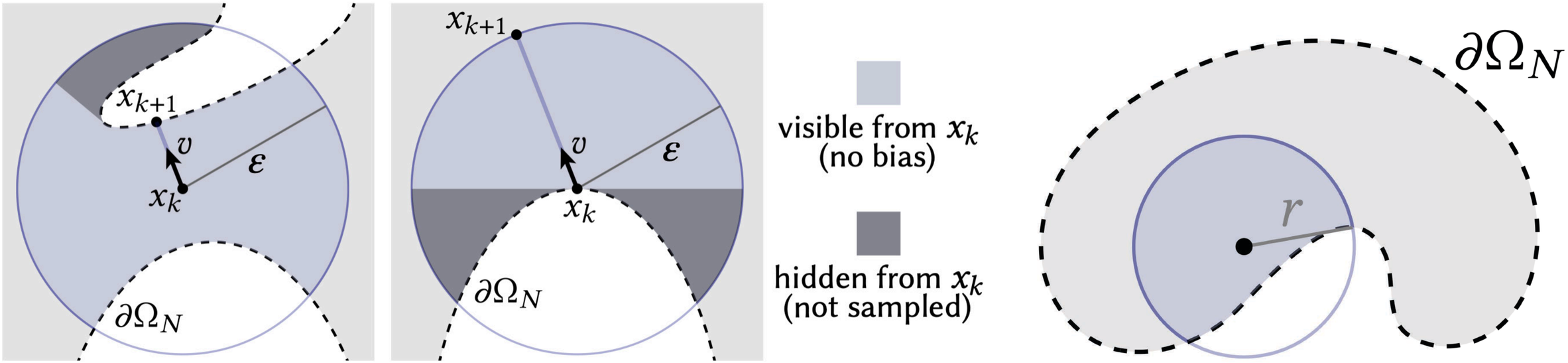
Stopping tolerance ε for Neumann & Robin boundaries

Star radius shrinks near **concave** parts of the Neumann boundary



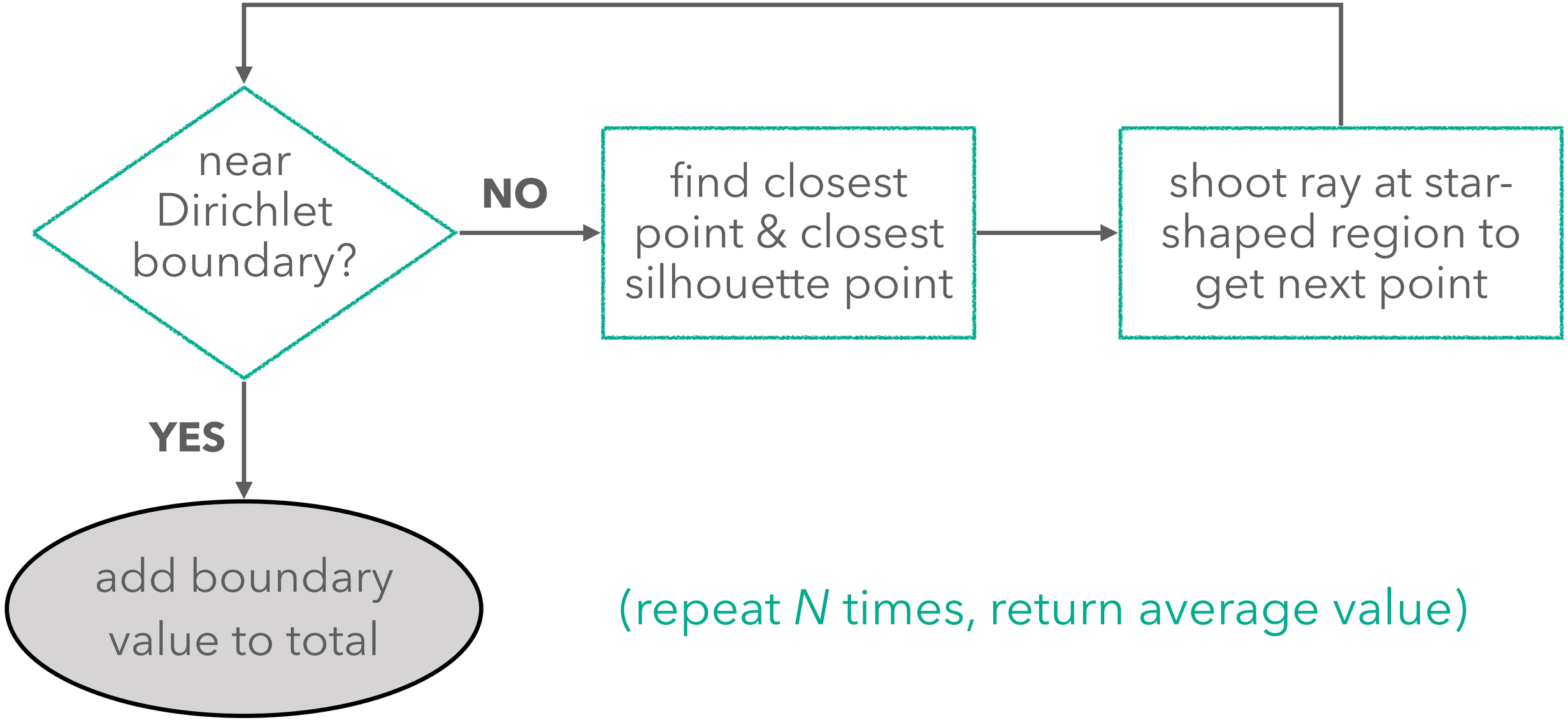
Stopping tolerance ε for Neumann & Robin boundaries

Star radius shrinks near **concave** parts of the Neumann boundary



Minimum radius parameter ε has a **performance vs bias tradeoff**

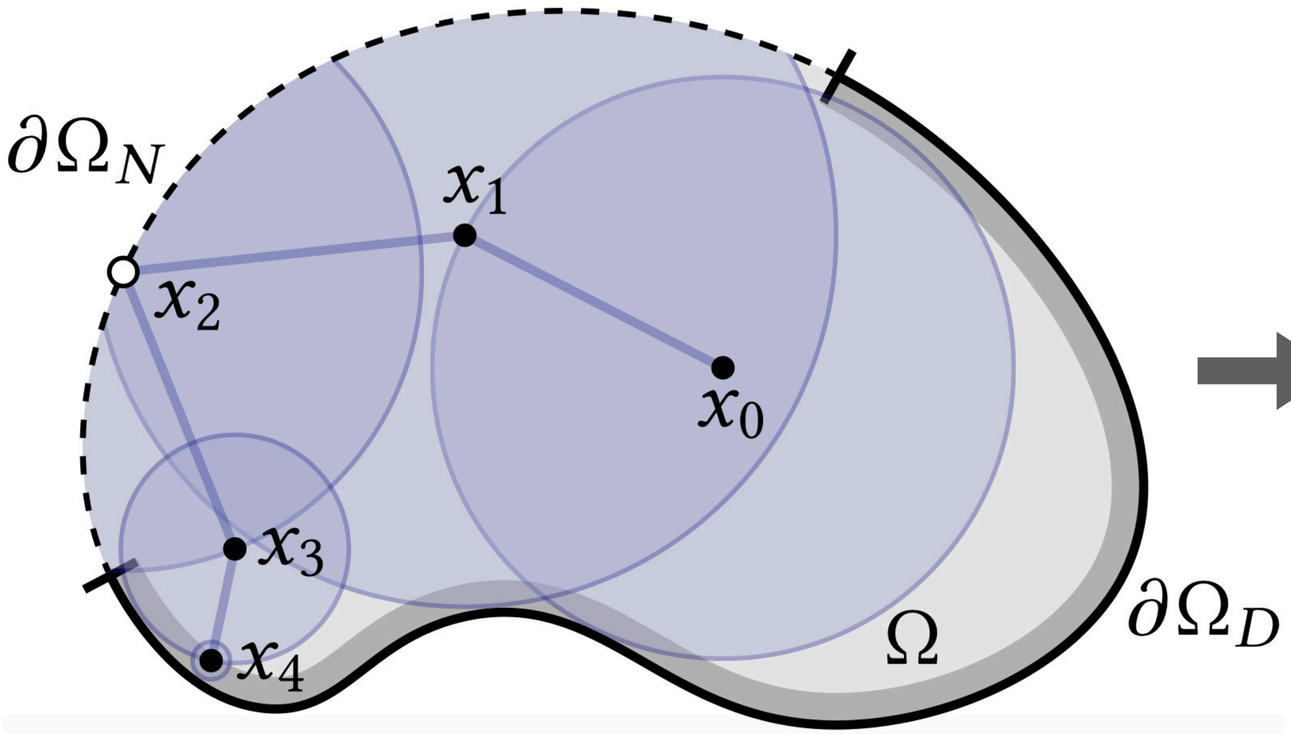
Walk on stars (Laplace, Dirichlet, zero-Neumann)



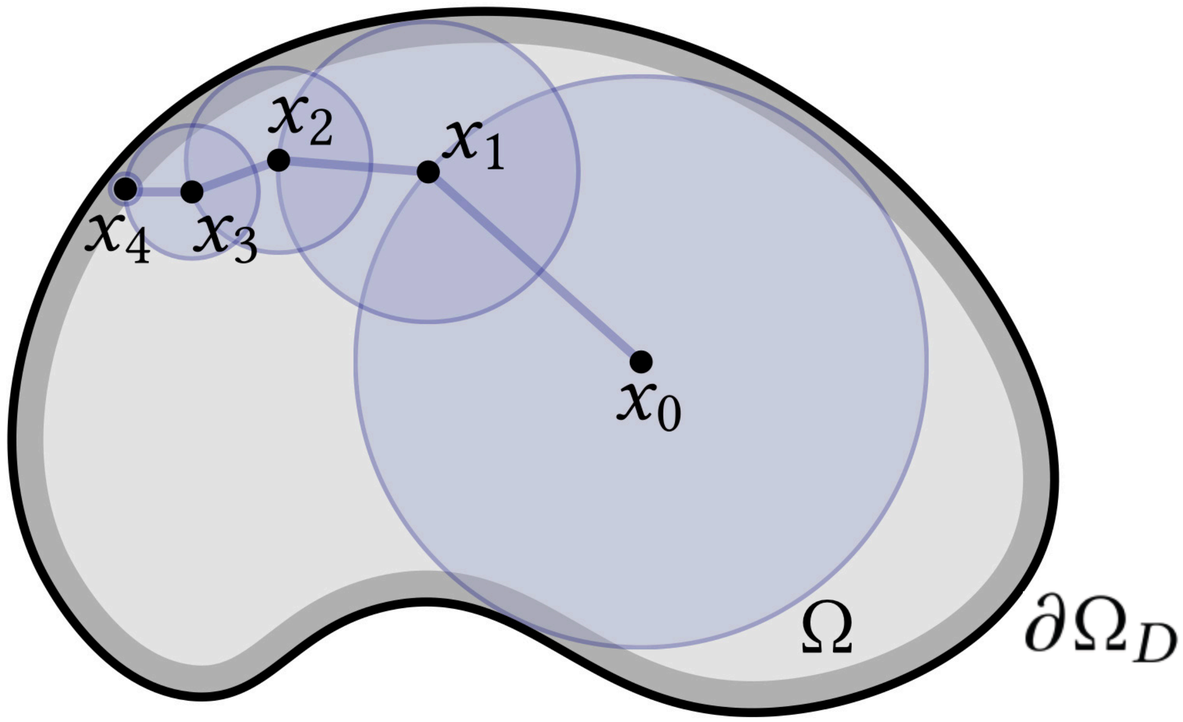
(repeat *N* times, return average value)

Walk on stars

Automatically **becomes WoS** when there is no Neumann boundary



mixed reflecting/absorbing



pure Dirichlet (absorbing)

PDE solver in 150 lines of code

standard C++ *(no external dependencies)*

```
#include <algorithm>
#include <array>
#include <complex>
#include <functional>
#include <random>
#include <vector>
#include <fstream>
using namespace std;

double random( double rMin, double rMax ) {
   double u = (double)rand()/(double)RAND_MA
   return u*(rMax-rMin) + rMin;
}

using Vec2D = complex<double>;
double length( Vec2D u ) { return sqrt( norm
double angleOf(Vec2D u) { return arg(u); }
Vec2D rotate90( Vec2D u ) { return Vec2D( -
double   dot(Vec2D u, Vec2D v) { return rea
double cross(Vec2D u, Vec2D v) { return rea

Vec2D closestPoint( Vec2D x, Vec2D a, Vec2D
   Vec2D u = b-a;
   double t = clamp( dot(x-a,u)/dot(u,u), 0.
   return (1.0-t)*a + t*b;
}

bool isSilhouette( Vec2D x, Vec2D a, Vec2D b
   return cross(b-a,x-a) * cross(c-b,x-b) <
}

double rayIntersection( Vec2D x, Vec2D v, Ve
   Vec2D u = b - a;
   Vec2D w = x - a;
   double d = cross(v,u);
   double s = cross(v,w) / d;
   double t = cross(u,w) / d;
   if (t > 0. && 0. <= s && s <= 1.) {
      return t;
   }
   return numeric_limits<double>::infinity()
}

using Polyline = vector<Vec2D>;

double distancePolylines( Vec2D x, const vec
   double d = numeric_limits<double>::infini
   for( int i = 0; i < P.size(); i++ ) { //
      for( int j = 0; j < P[i].size()-1; j++
         Vec2D y = closestPoint( x, P[i][j],
```

```
d = min( d, length(x-y) ); // update minimum distance
```

```
Vec2D n{ 0.0, 0.0 }; // assume x0 is an interior point, and has no normal
 whether x is on the interior or boundary
e; // radii used to define star shaped region

 the Dirichlet boundary
s( x, boundaryDirichlet );
ncePolylines( x, boundaryNeumann );
t, dSilhouette ));
 M_PI );
from a hemisphere around the normal
(n);

a) }; // unit ray direction
 r, boundaryNeumann, n, onBoundary );

< maxSteps);
ibution of the boundary value

stimate

 floor(8.0*real(x)), 2.0 ); }

, Vec2D(1.0, 0.2) }},
, Vec2D(0.2, 1.0) }}

, Vec2D(1.0, 1.0) }},
, Vec2D(0.2, 0.2) }}

uble)s),
uble)s) );

let, boundaryNeumann, lines );
```

https://geometry.cs.cmu.edu/stars

Walk on Stars
(Implementation Guide)

Rohan Sawhney, Bailey Miller, Ioannis Gkioulekas, Keenan Crane

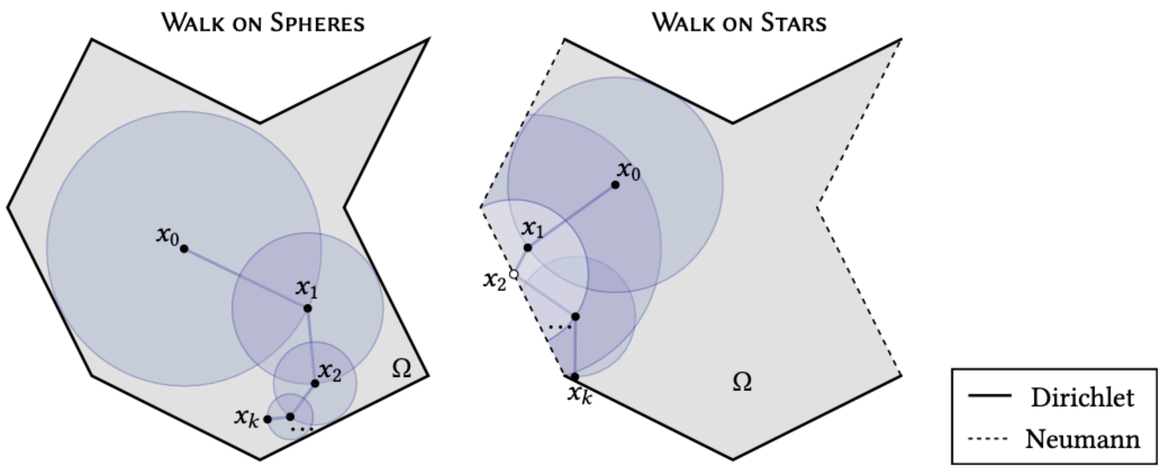


Figure 1: Walks taken by the WoS algorithm *(left)* and the WoSt algorithm *(right)*. The algorithms are virtually identical, except that WoSt replaces spheres with star-shaped regions, enabling walks to reflect off Neumann boundaries.

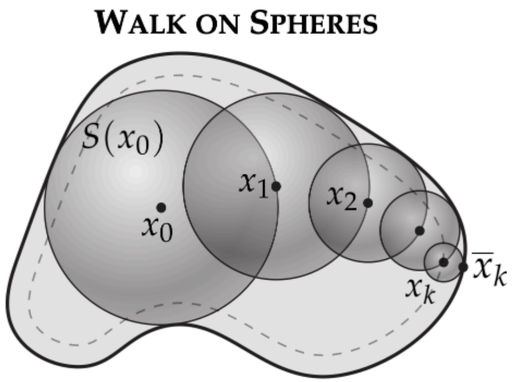
This note provides a step-by-step tutorial on how to implement the *walk on stars (WoSt)* algorithm of Sawhney et al.

Reference implementation

README.md

ZOMBIE

Zombie is a C++ header-only library for solving fundamental partial differential equations (PDEs) like the Poisson equation using the *walk on spheres (WoS)* method and its *extensions*. Unlike finite element, boundary element, or finite difference methods, WoS does not require a volumetric grid or mesh, nor a high-quality boundary mesh. Instead, it uses random walks and the Monte Carlo method to solve the problem directly on the original boundary representation. It can also provide accurate solution values at a single query point, rather than needing to solve the problem over the entire domain. This *talk* provides an overview of WoS, while the following papers discuss its present capabilities in greater detail:



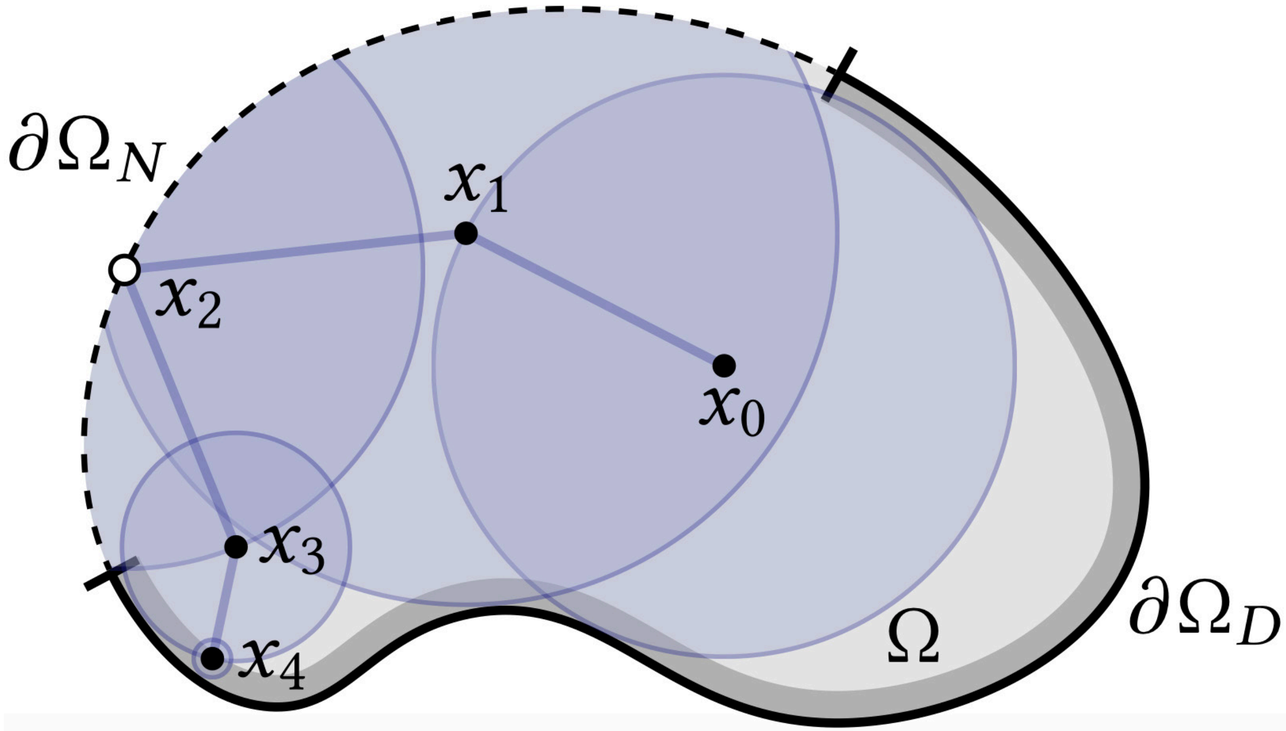
> *Monte Carlo Geometry Processing:A Grid-Free Approach to PDE-Based Methods on Volumetric Domains*
> *Walk on Stars: A Grid-Free Monte Carlo Method for PDEs with Neumann Boundary Conditions*
> *Grid-Free Monte Carlo for PDEs with Spatially Varying Coefficients*
> *Boundary Value Caching for Walk on Spheres*

Additional features – non-zero Neumann conditions

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

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

$$\frac{\partial u}{\partial n} = h \quad \text{on } \partial\Omega_N$$



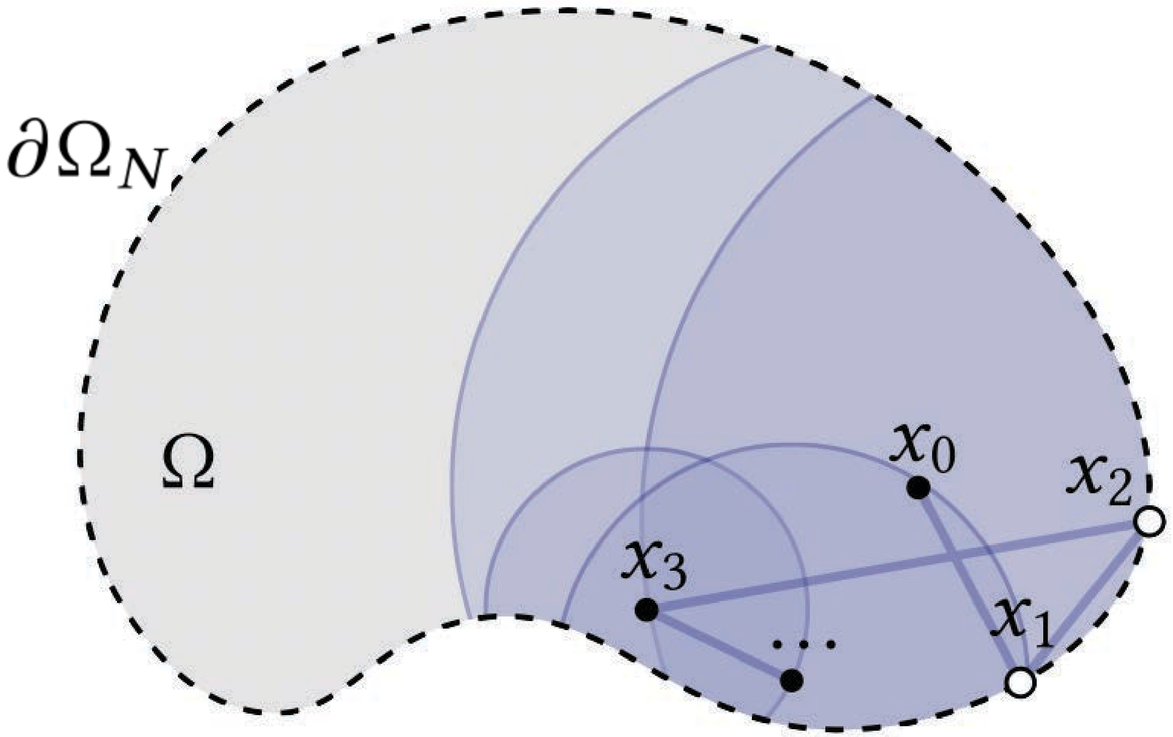
walk picks up contribution with every reflection on Neumann boundary

Additional features – pure Neumann problems

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

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

$$\frac{\partial u}{\partial n} = h \quad \text{on } \partial\Omega_N$$



walk continues forever…

use **Tikhonov regularization**

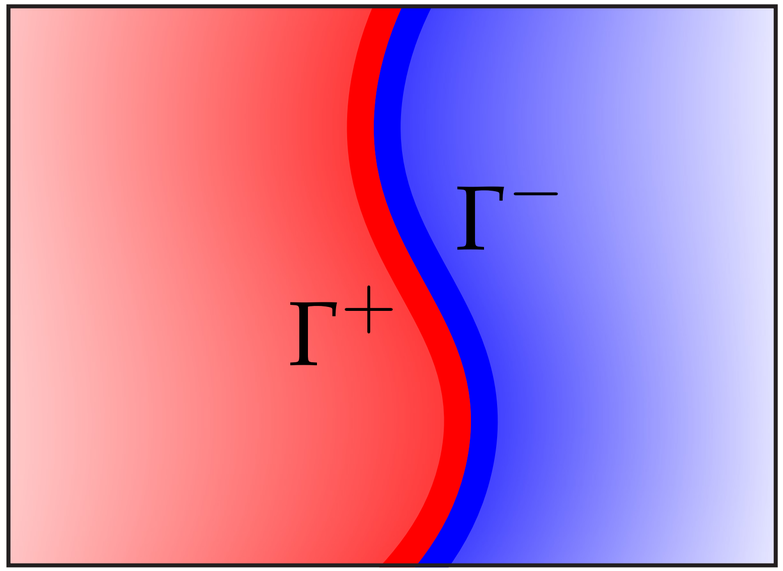
to terminate walks!

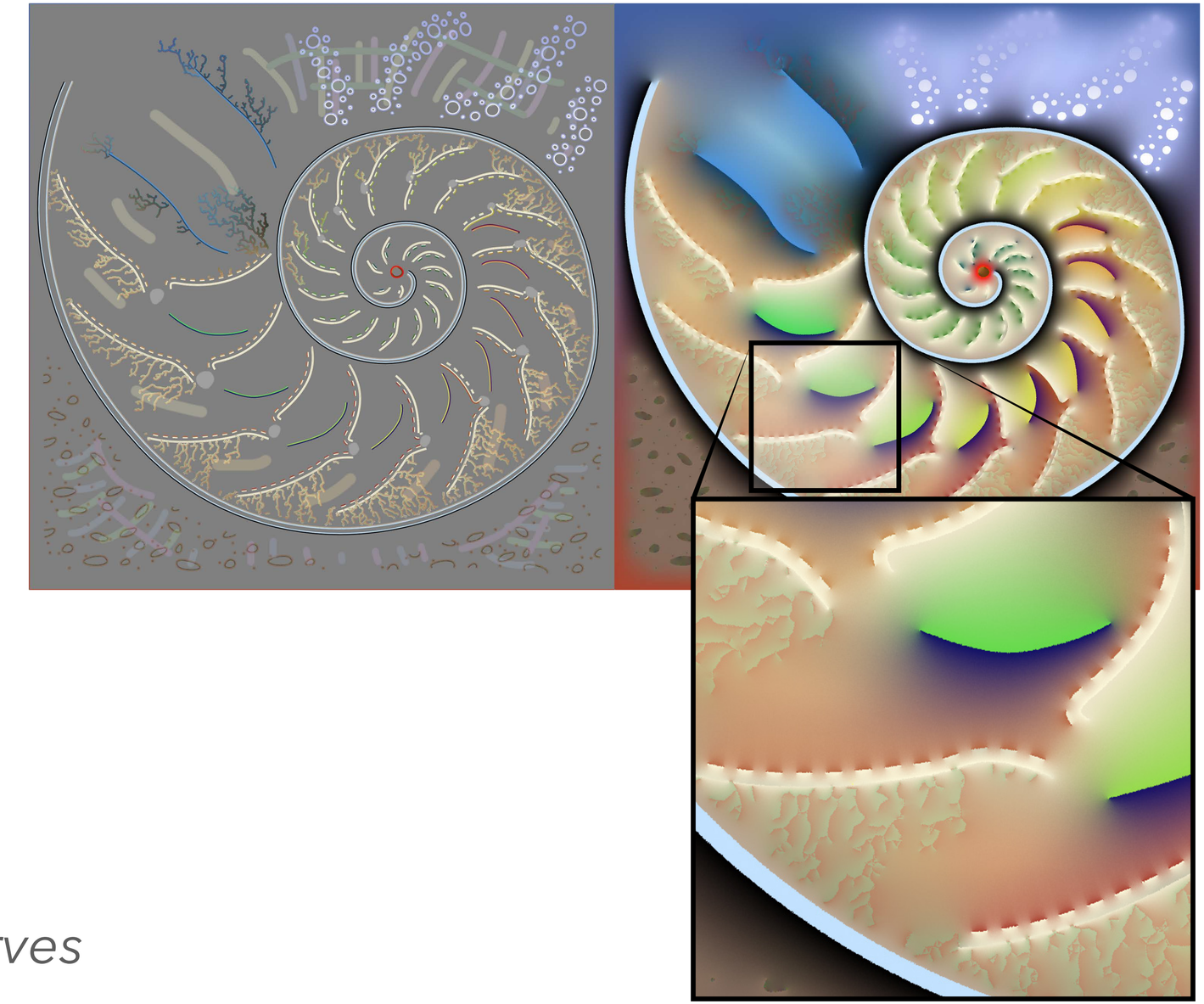
Additional features – Open domains & double-sided conditions

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

$$u = g^{+} \quad \text{on } \Gamma^{+}$$

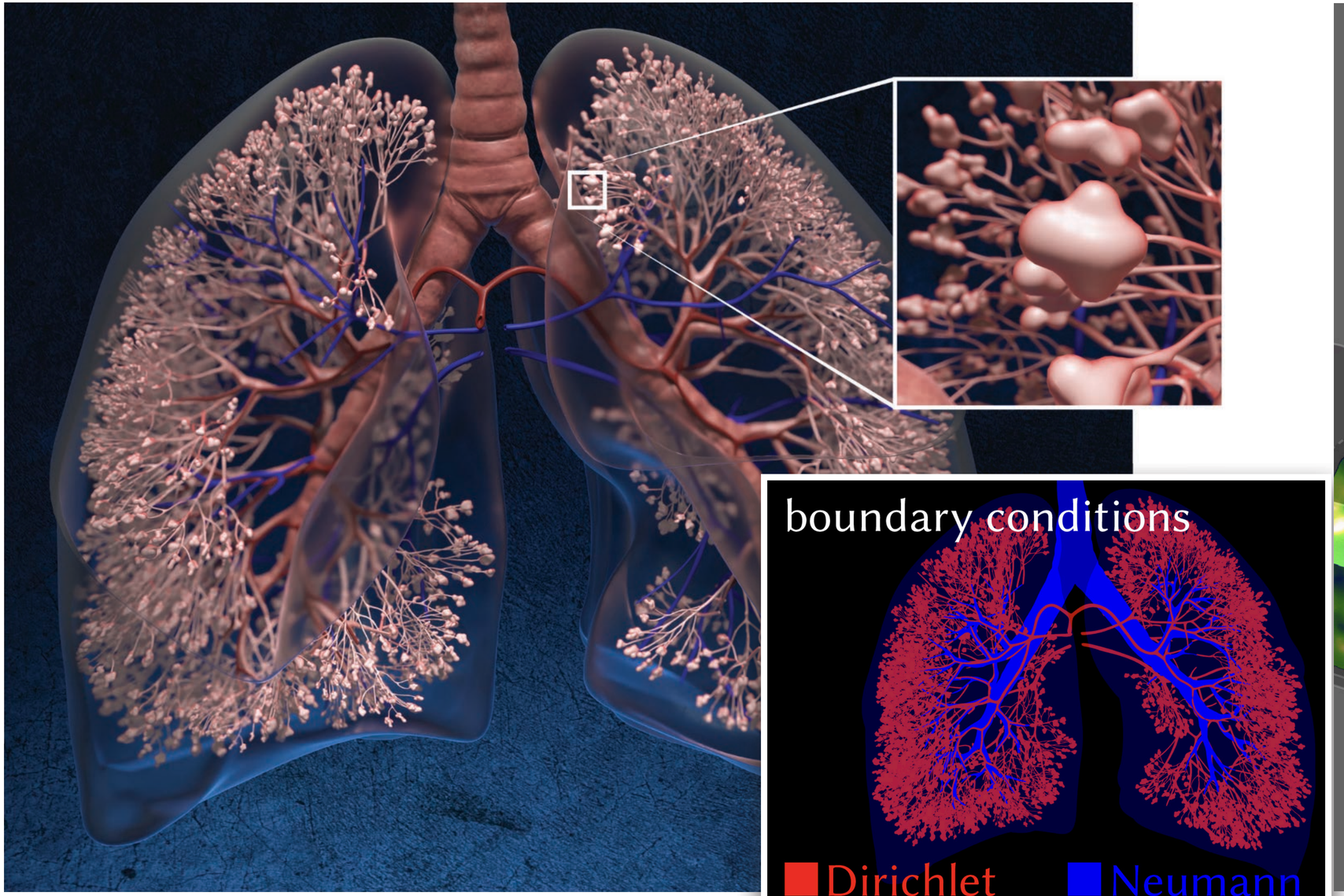
$$u = g^{-} \quad \text{on } \Gamma^{-}$$

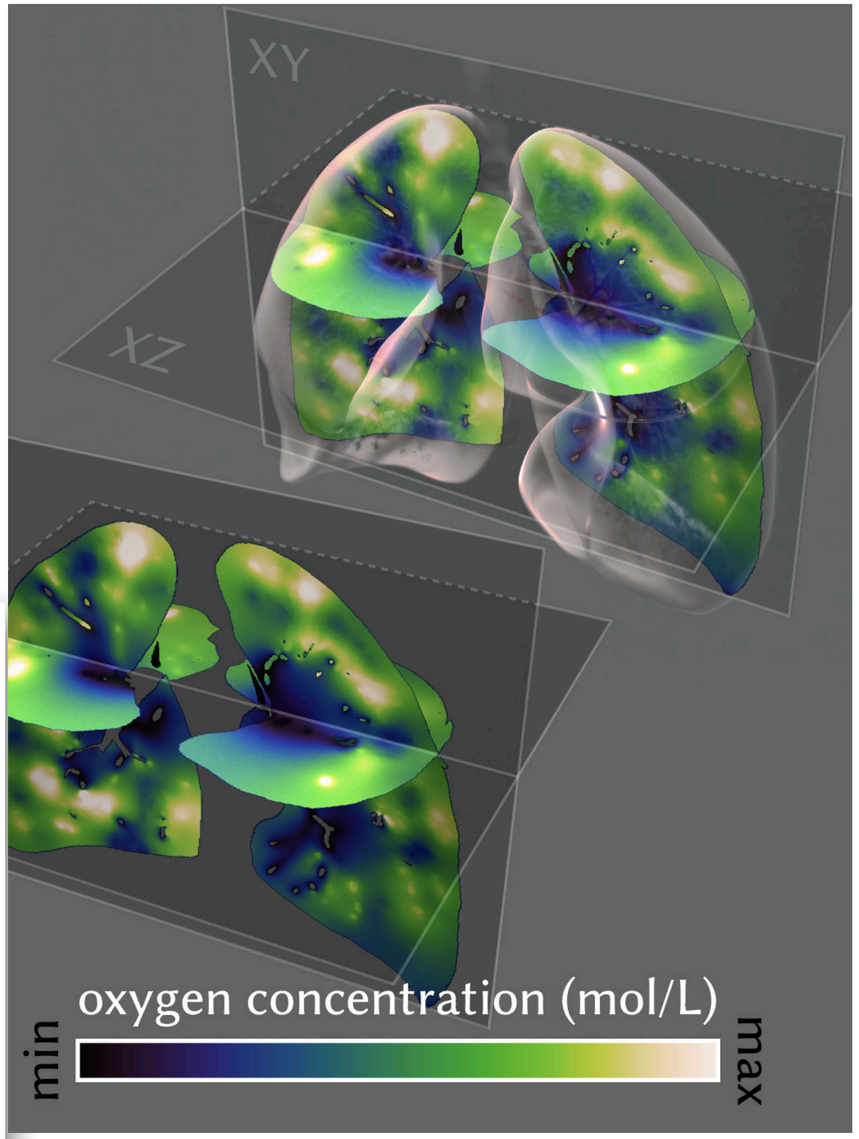




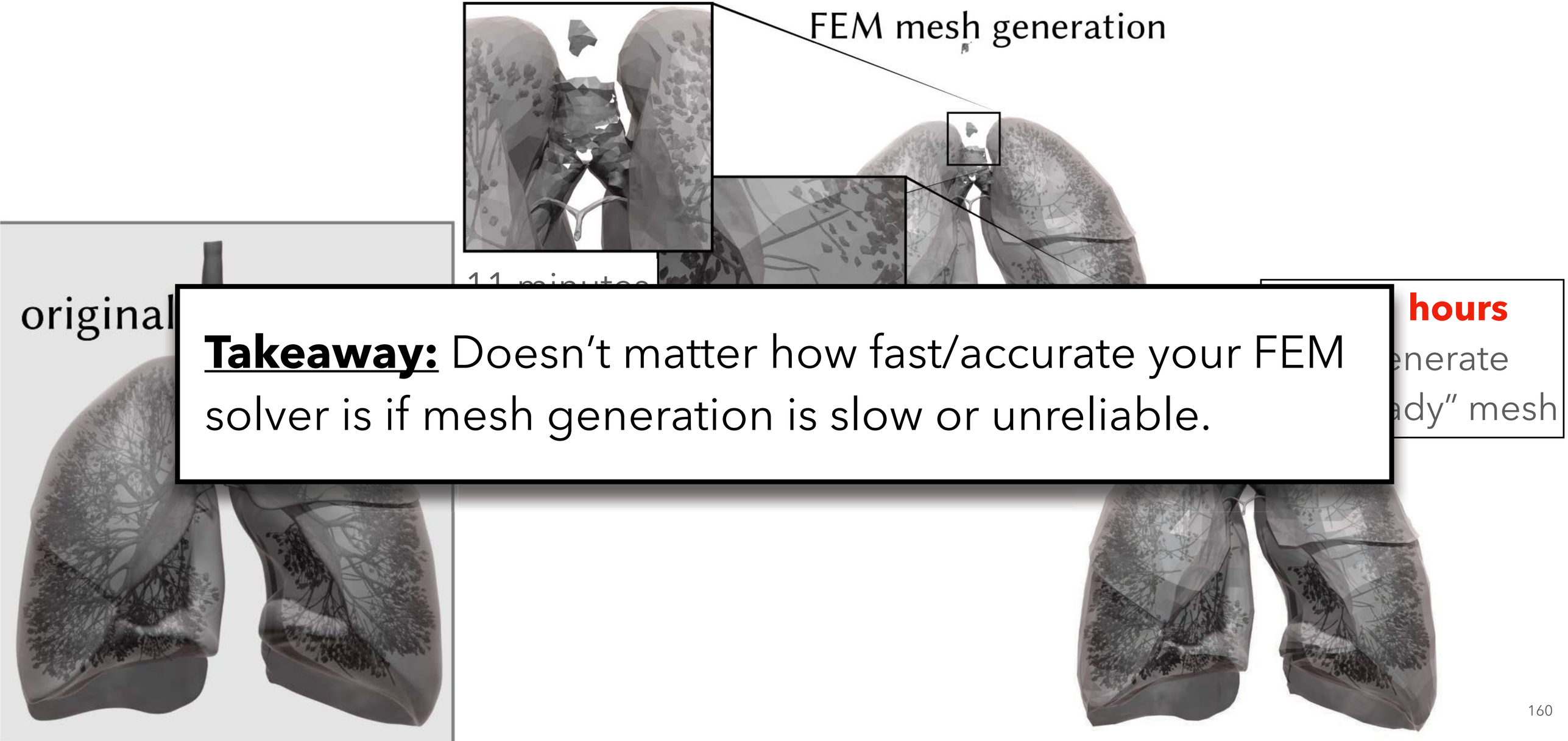
Γ – *collection of open & closed curves*

Oxygen diffusion – walk on stars



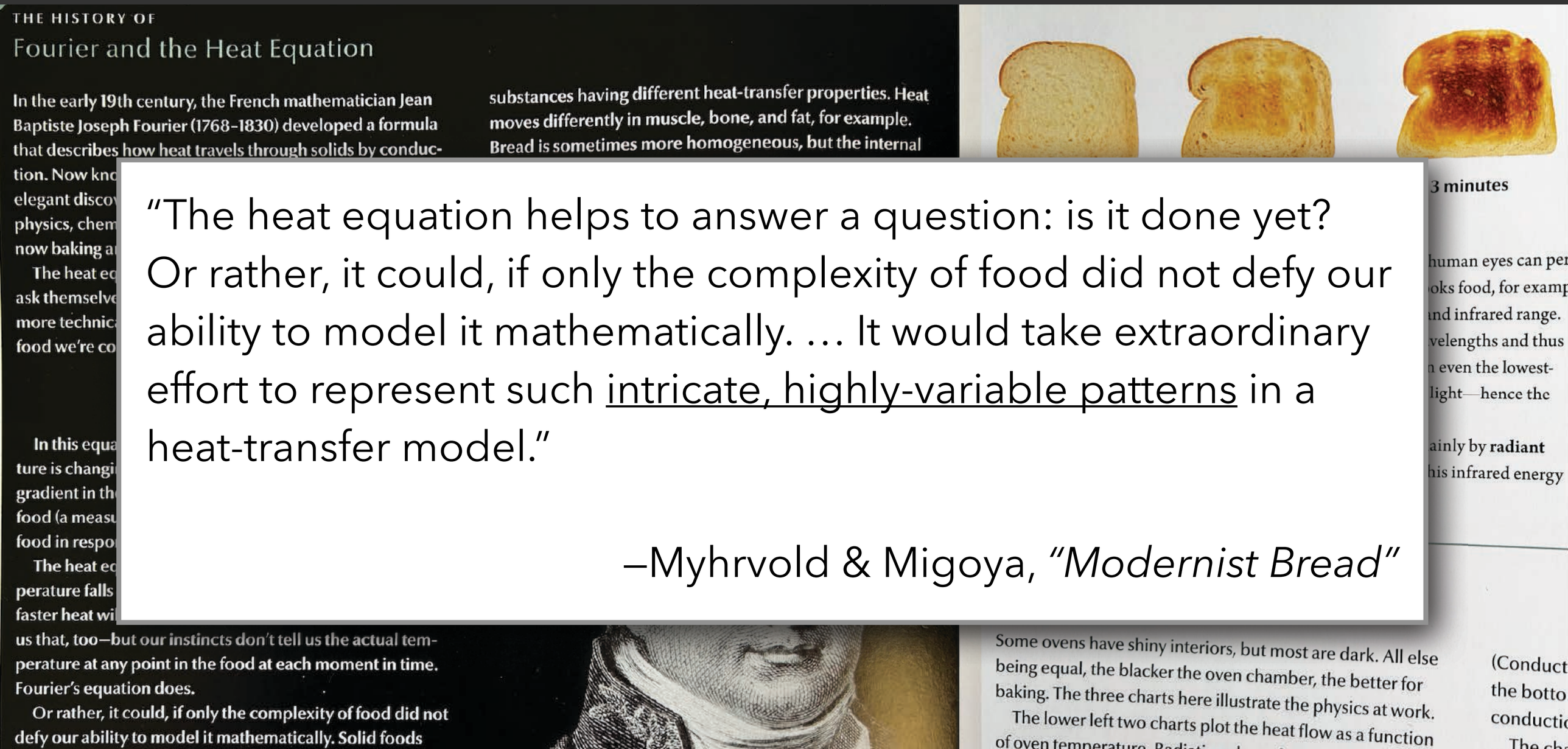


Oxygen diffusion – FEM



<u>Takeaway:</u> Doesn't matter how fast/accurate your FEM solver is if mesh generation is slow or unreliable.

Thermal transfer



THE HISTORY OF

Fourier and the Heat Equation

In the early 19th century, the French mathematician Jean Baptiste Joseph Fourier (1768–1830) developed a formula that describes how heat travels through solids by conduction. Now kno
elegant disco
physics, chem
now baking a
The heat eq
ask themselve
more technic
food we're co

In this equa
ture is changi
gradient in th
food (a measu
food in respo
The heat eq
perature falls
faster heat wi
us that, too—but our instincts don't tell us the actual temperature at any point in the food at each moment in time. Fourier's equation does.

Or rather, it could, if only the complexity of food did not defy our ability to model it mathematically. Solid foods

substances having different heat-transfer properties. Heat moves differently in muscle, bone, and fat, for example. Bread is sometimes more homogeneous, but the internal

3 minutes

human eyes can per
oks food, for examp
nd infrared range.
velengths and thus
n even the lowest-
light—hence the

ainly by **radiant**
his infrared energy

Some ovens have shiny interiors, but most are dark. All else being equal, the blacker the oven chamber, the better for baking. The three charts here illustrate the physics at work.

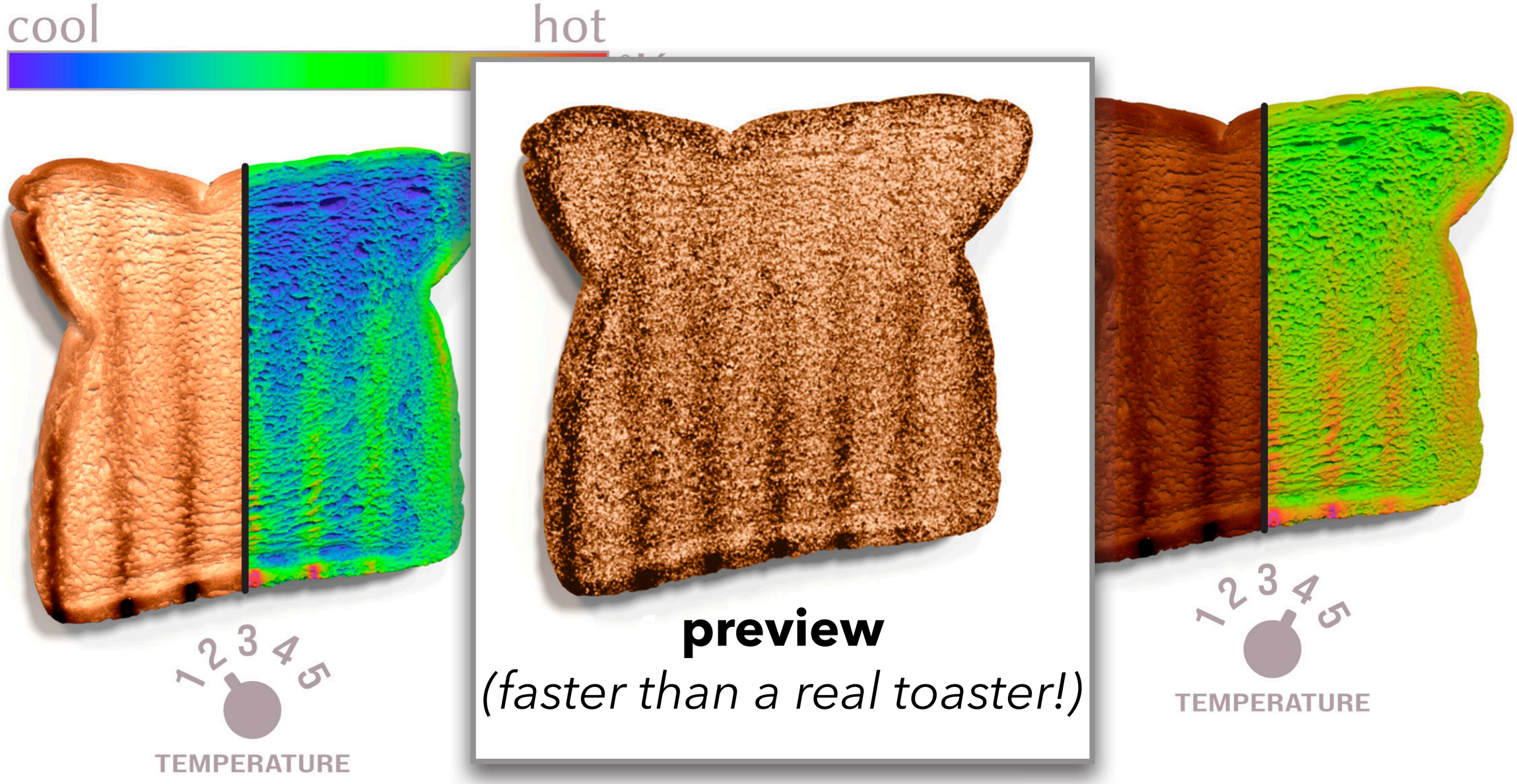
The lower left two charts plot the heat flow as a function of oven temperature. Radiati

(Conduct
the botto
conductio

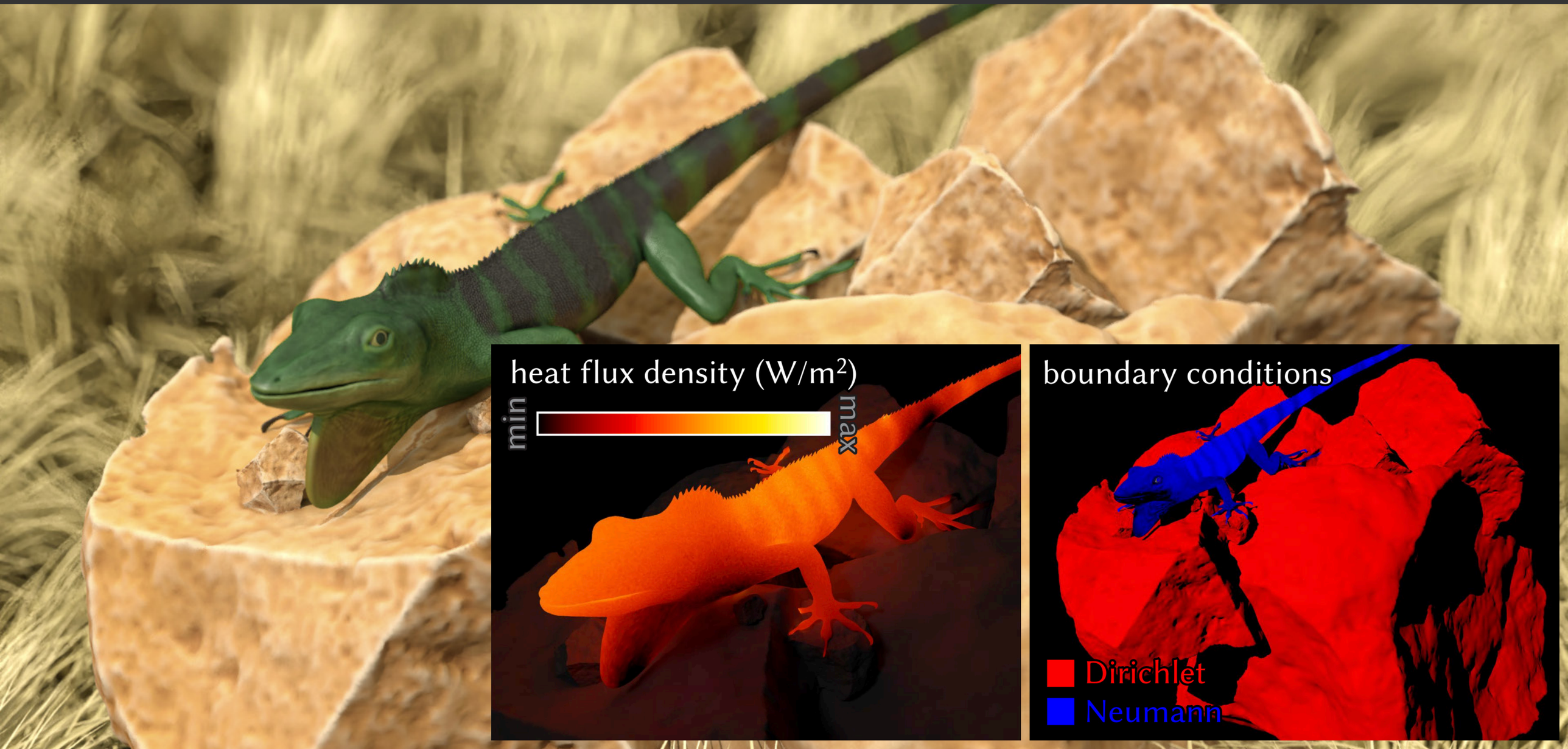
"The heat equation helps to answer a question: is it done yet? Or rather, it could, if only the complexity of food did not defy our ability to model it mathematically. … It would take extraordinary effort to represent such <u>intricate, highly-variable patterns</u> in a heat-transfer model."

–Myhrvold & Migoya, *"Modernist Bread"*

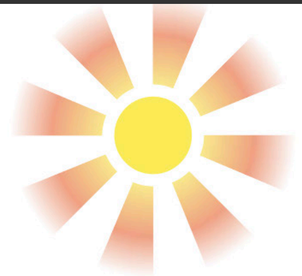
Thermal transfer on a detailed CT scan (4 million triangles)



Thermal transfer

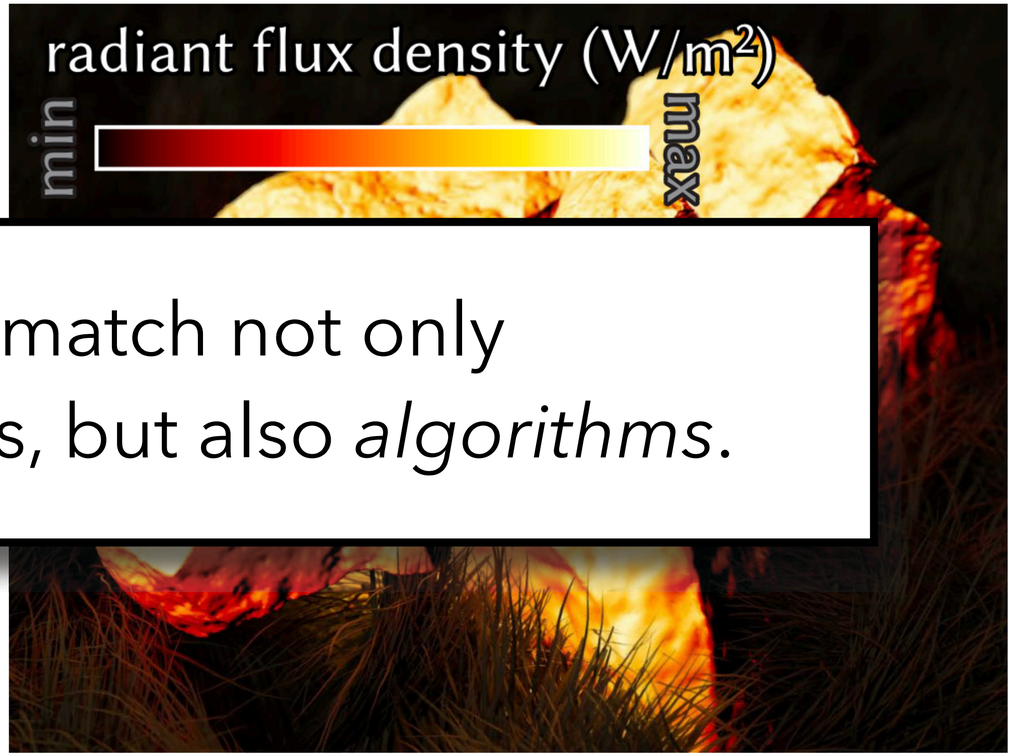


Thermal transfer

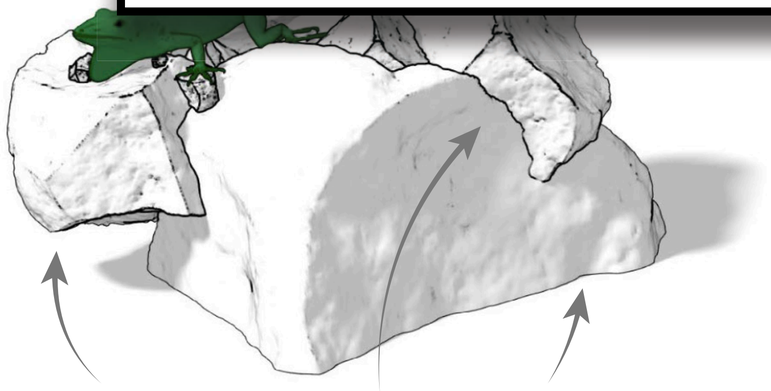


radiative heating
(ray tracing)

heat exch
(walk on s



Takeaway: Easy to mix & match not only geometric representations, but also *algorithms*.



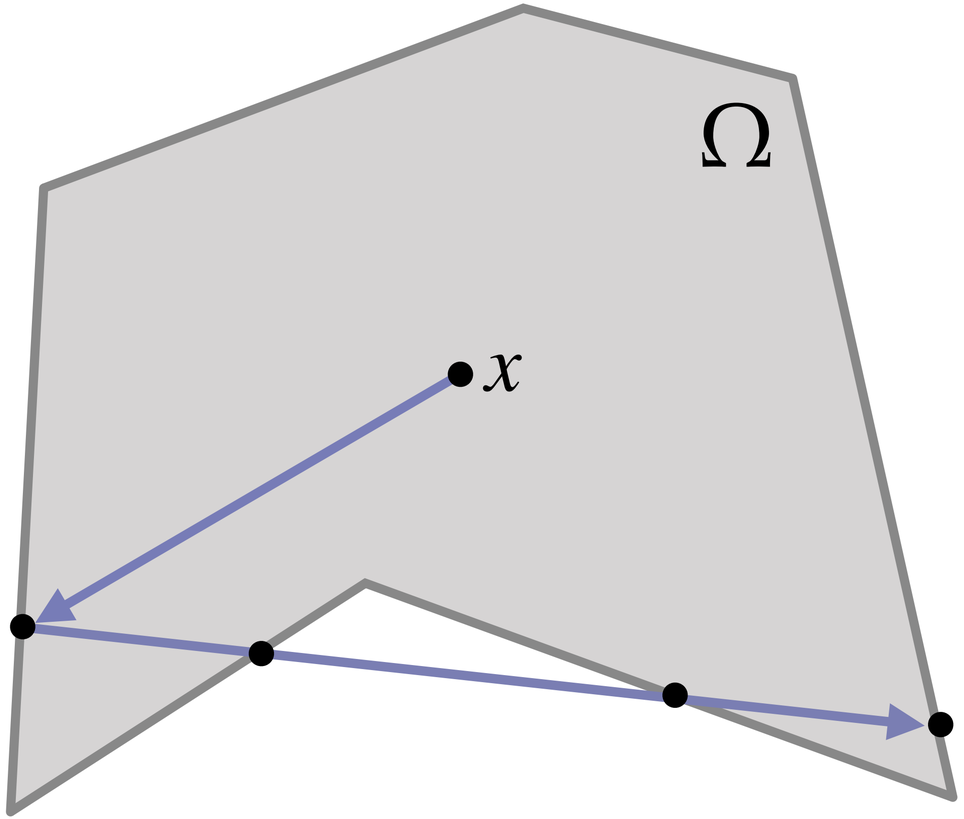
signed distance functions

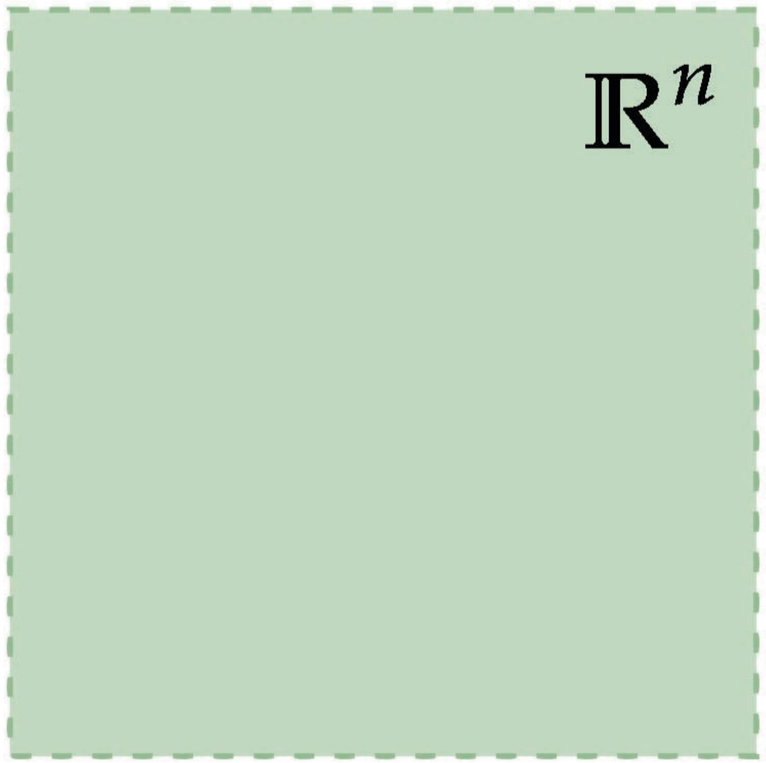
boundary conditions evaluated "on demand"

Walk on boundary [Sabelfeld & Simonov 2013]

A = input domain

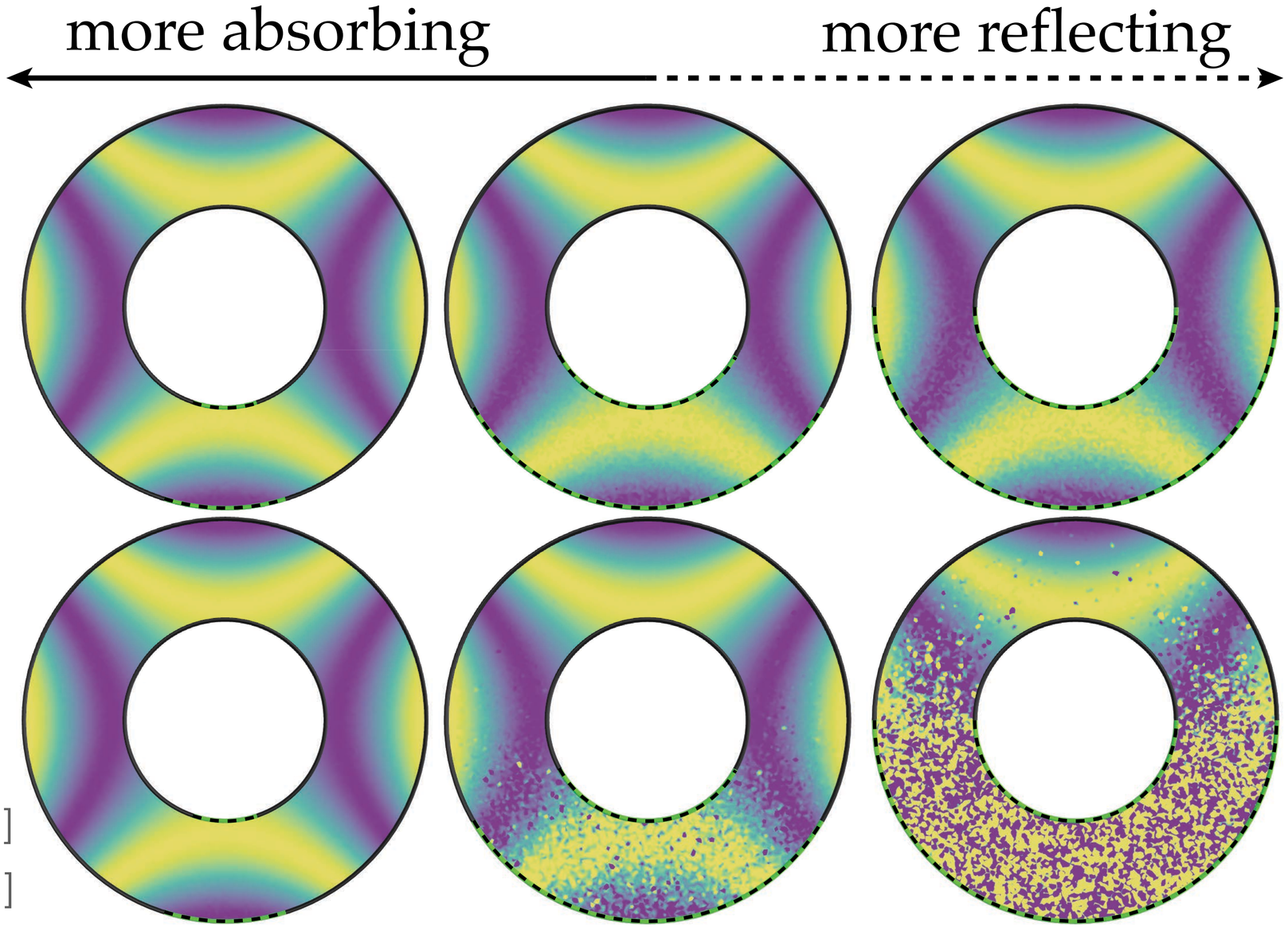
C = Euclidean space





Multiple intersections $\Longrightarrow$ ***exponential* variance**

Comparison to walk on boundary



Walk on Stars

Walk on Boundary

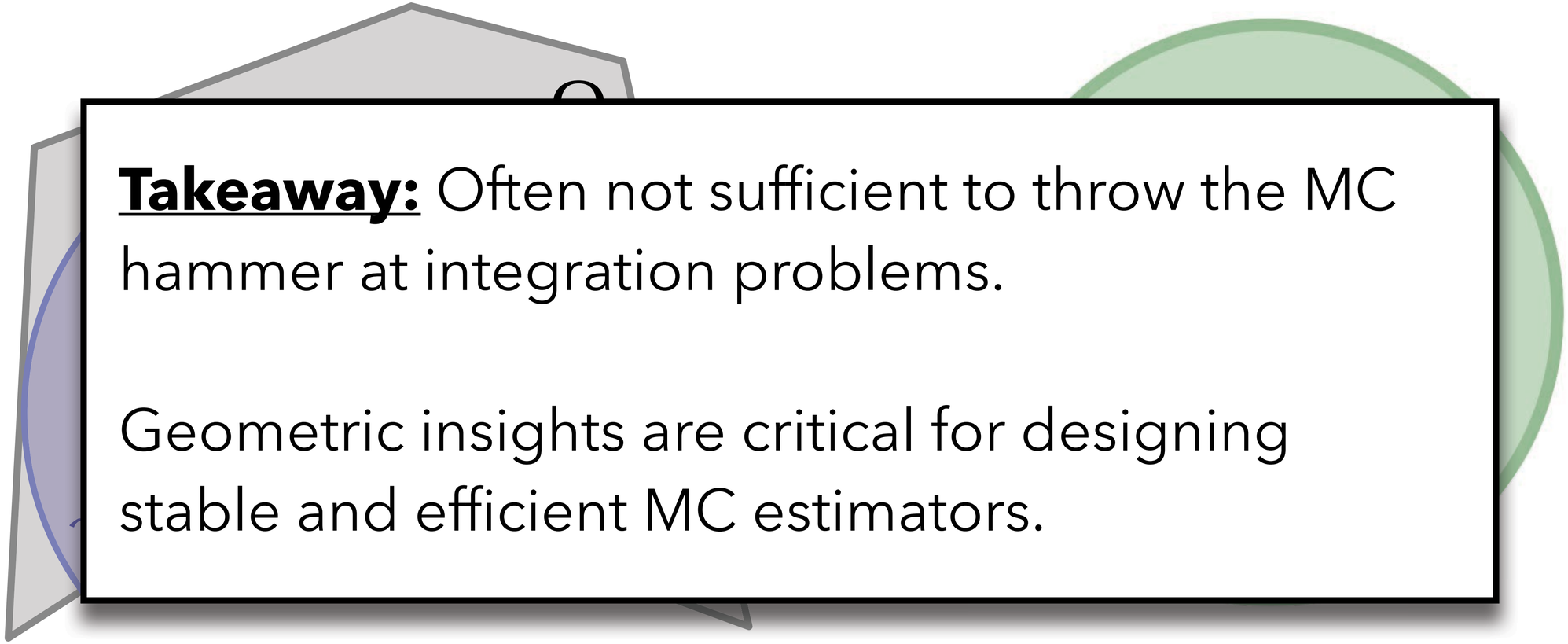
[Sabelfeld & Simonov 2013]

[Sugimoto et al 2023]

Walk on stars

A = star-shaped region

C = ball



<u>Takeaway:</u> Often not sufficient to throw the MC hammer at integration problems.

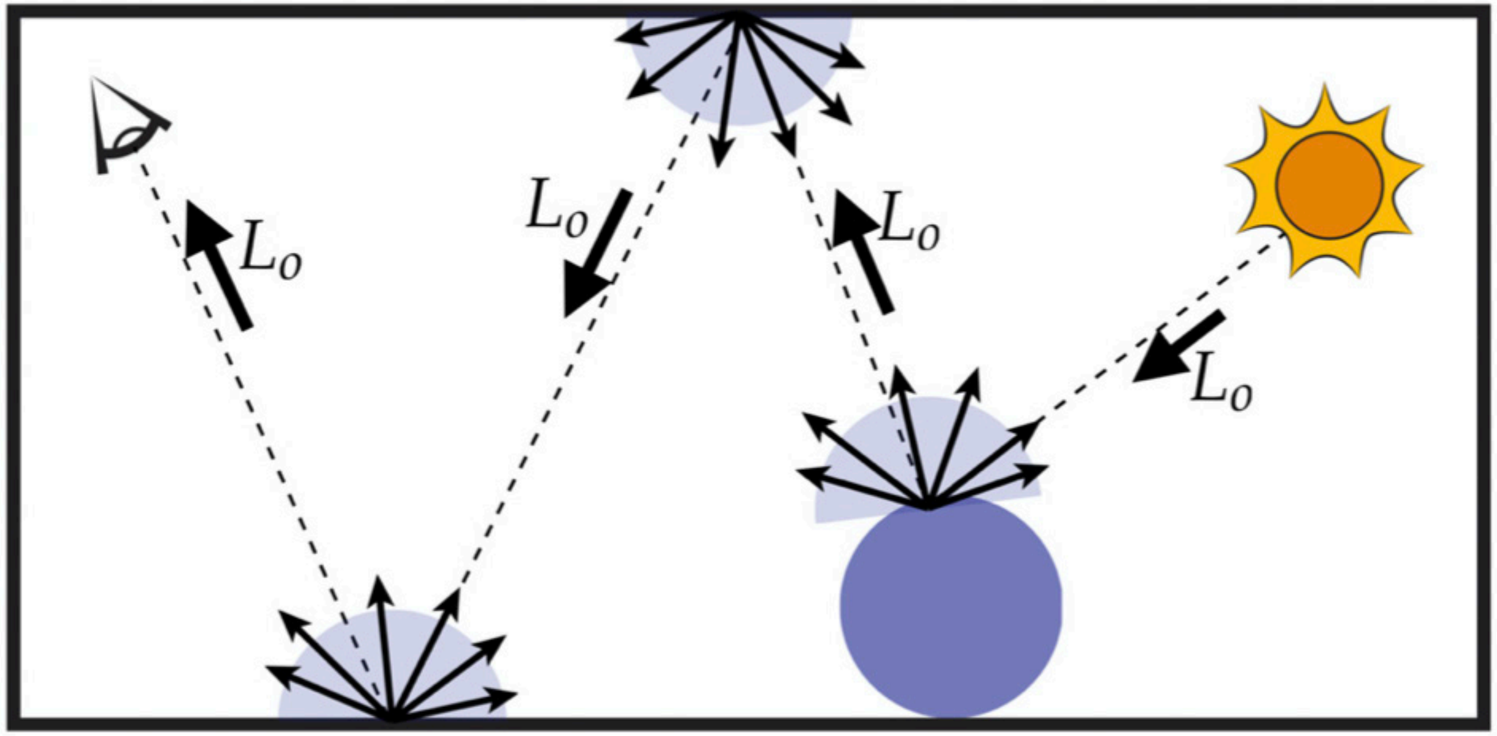
Geometric insights are critical for designing stable and efficient MC estimators.

What other subdomains can we use?

Sampling signed solid angle in photorealistic rendering

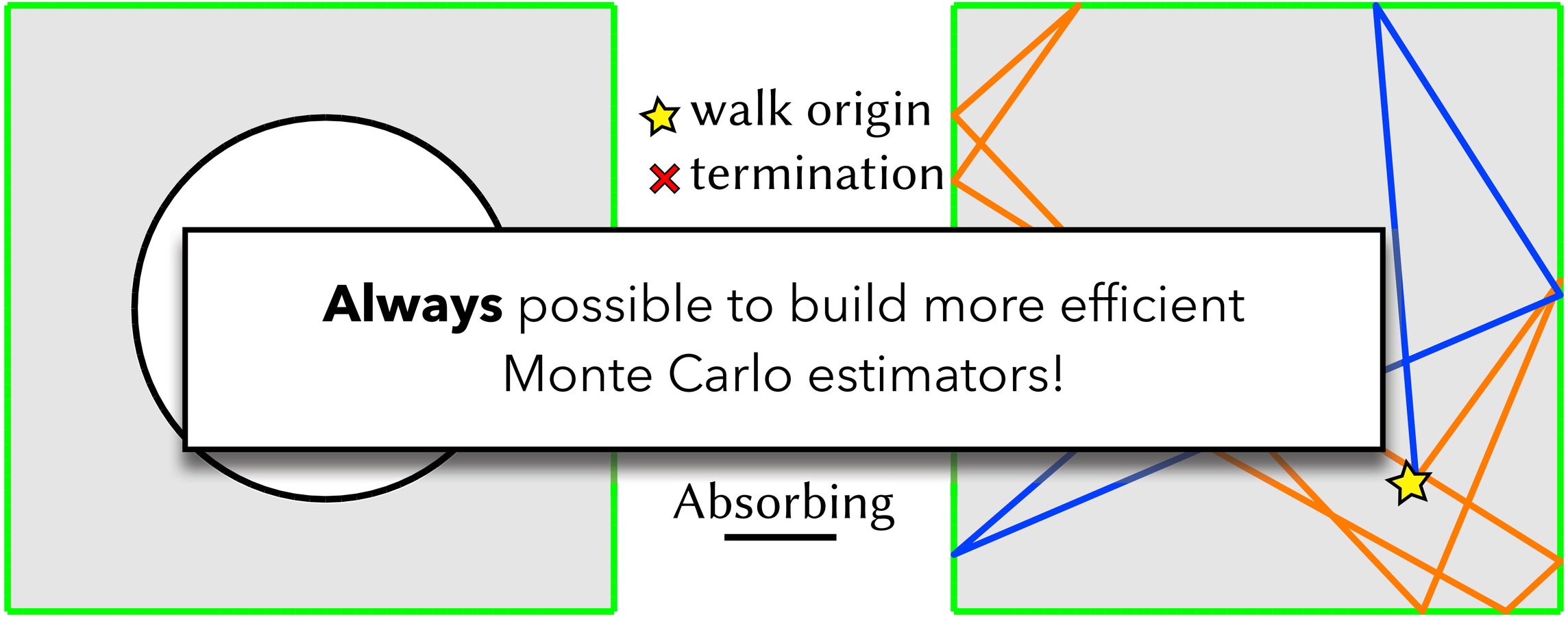
$$L_o(x) = \int_\Omega \underbrace{\rho(x,y)}_{\text{brdf}} \underbrace{L_i(y)}_{\text{radiance}} \underbrace{V(x,y)}_{\text{visibility}} \underbrace{\frac{n_y \cdot \widehat{y-x}}{4\pi r^2}}_{\text{geometry}} dA$$

sample y by tracing ray & returning first (visible) hit!



Critical to **importance sample** both V and $\frac{n_y \cdot \widehat{y-x}}{4\pi r^2}$

Long walk lengths



Always possible to build more efficient Monte Carlo estimators!

Ray Tracing Avg # Steps: 240

WoSt Avg # Steps: 219

PART 5:
WoS as Simulation of Brownian Motion

A tale of three types of equations…

differential equations

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

integral equations

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

stochastic equations

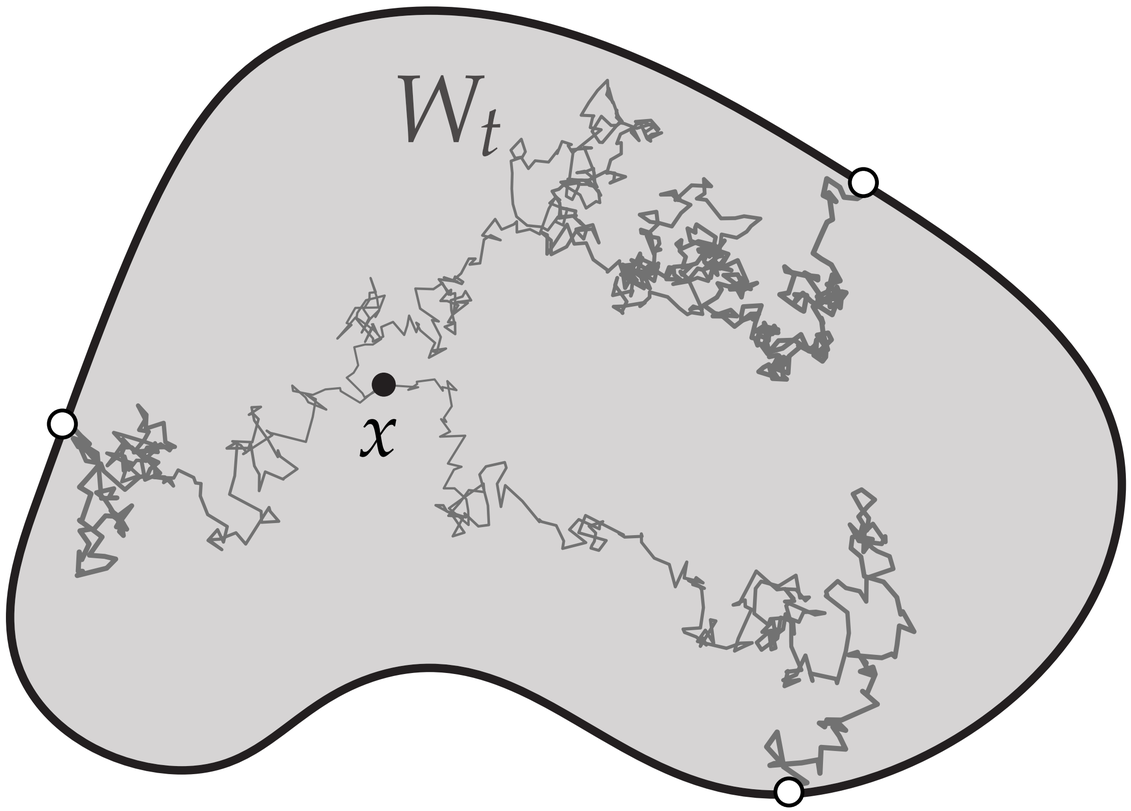
$$u(x) = \mathbb{E}\left[\int_0^\tau f(W_t)\, dt + g(W_\tau) \,\middle|\, W_0 = x \right]$$

Kakutani's Principle

Solution to Laplace eq can be computed by simulating **Brownian motion**

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

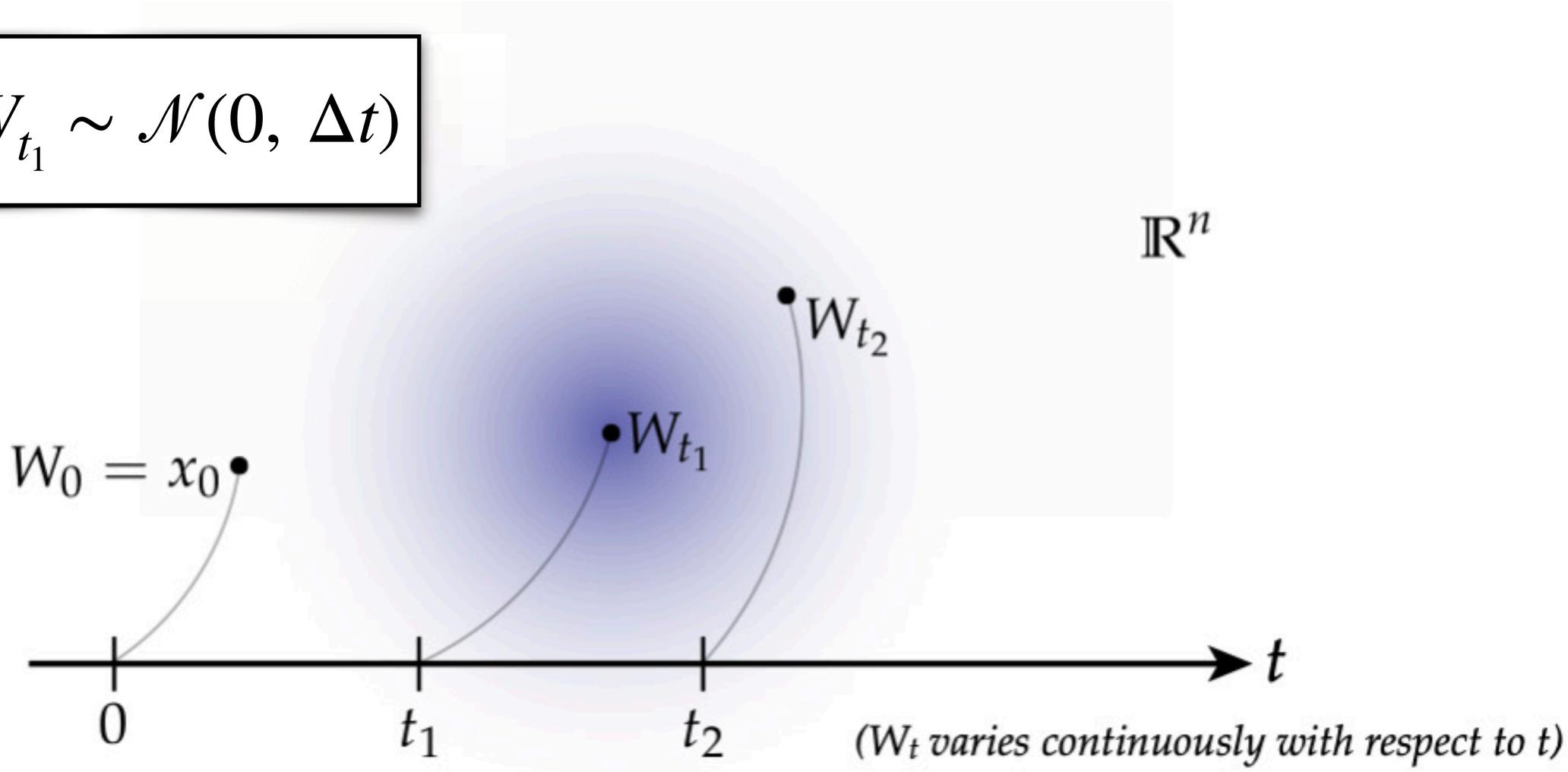
$$u(x) = \mathbb{E}[g(W_T)]$$
$$\approx \frac{1}{N}\sum_{i=1}^{N} g(W_T^i)$$



Brownian motion

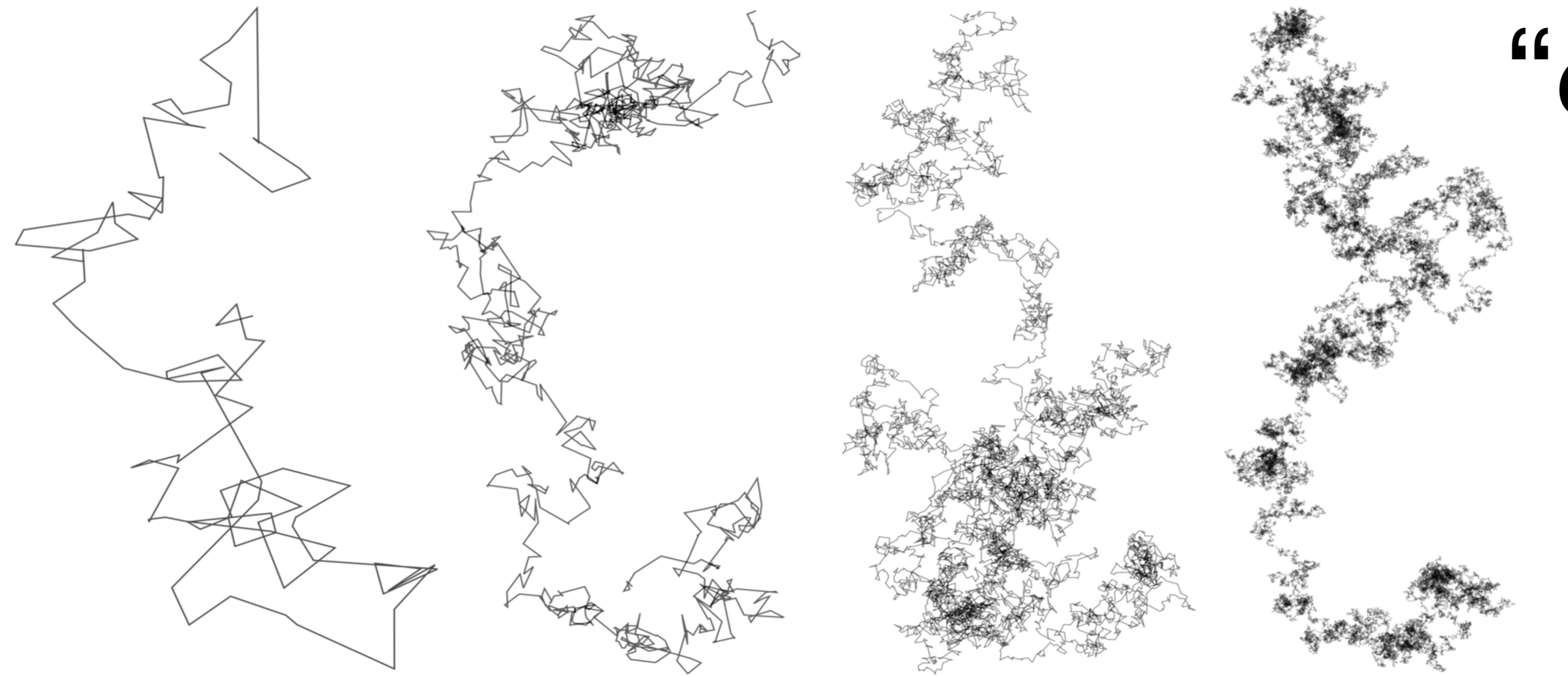
Collection of **independent normally-distributed increments**

$$W_{t_2} - W_{t_1} \sim \mathcal{N}(0,\ \Delta t)$$



Brownian motion

Collection of **independent normally-distributed increments**

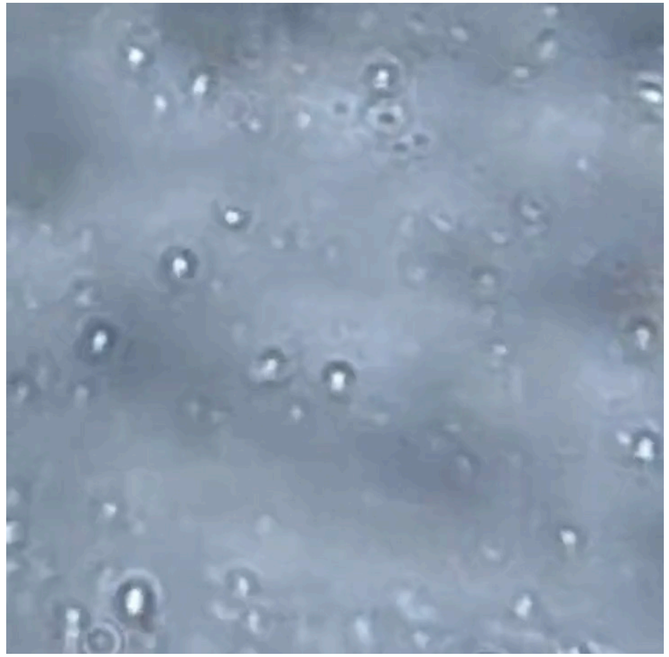


"$\mathrm{d}W_t$"

decreasing Δt

History of Brownian motion

Robert Brown (botanist)
studied erratic movement of pollen in water under a microscope (1827)

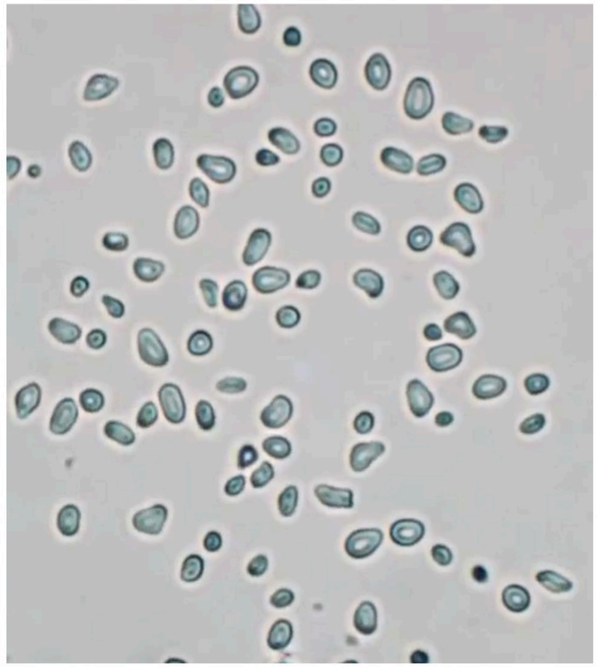


Albert Einstein (physicist)
related density of Brownian particles to solution of a heat equation (1905)

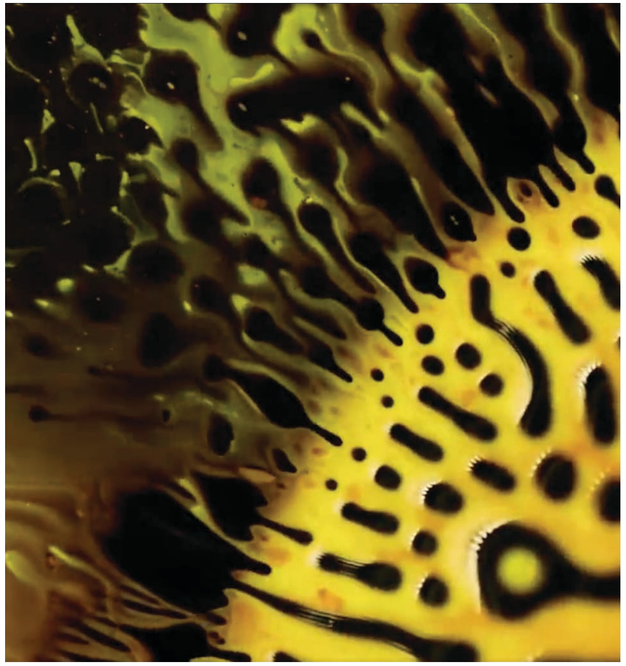
Nobert Wiener (mathematician)
rigorous mathematical theory of Brownian motion as continuous, non-differentiable paths (*Wiener process*, 1923)

Universality of Brownian motion

Even though random processes in nature, science, technology have *very* different origins, their aggregate behavior is well-predicted by BM

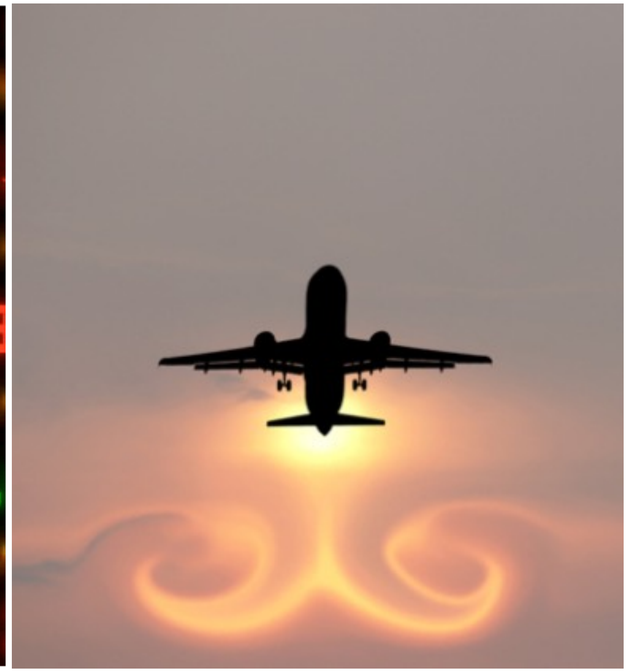


thermal fluctuations



reaction diffusion

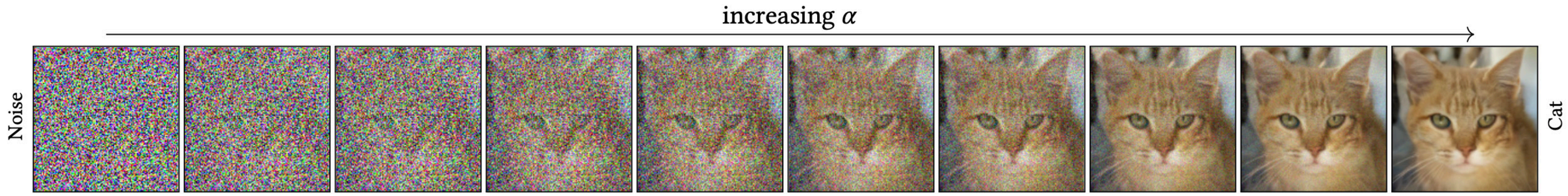
market volatility

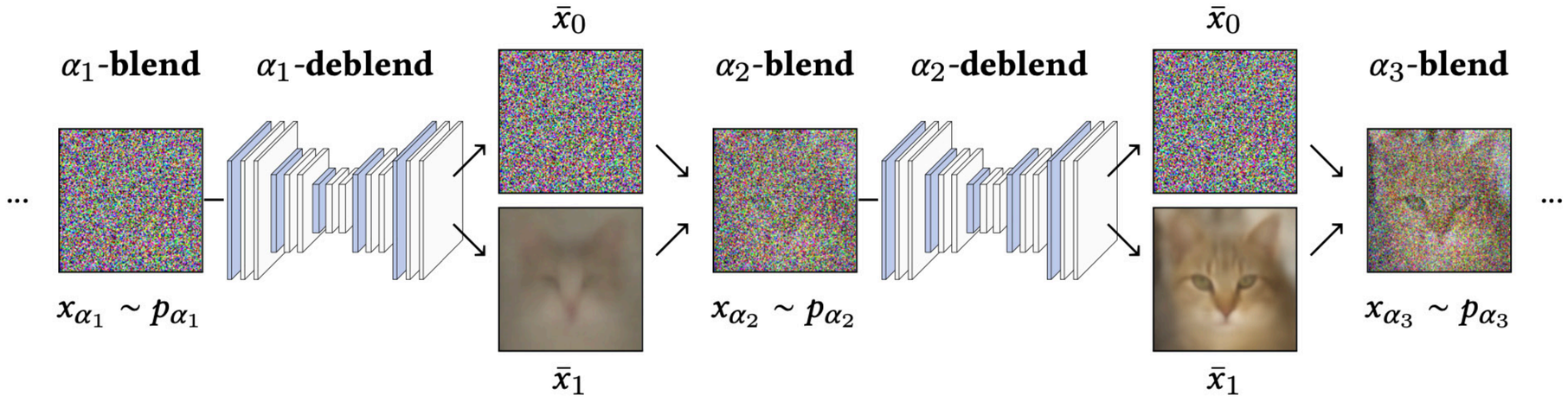


optimal control

Universality of Brownian motion

Even though random processes in nature, science, technology have *very* different origins, their aggregate behavior is well-predicted by BM



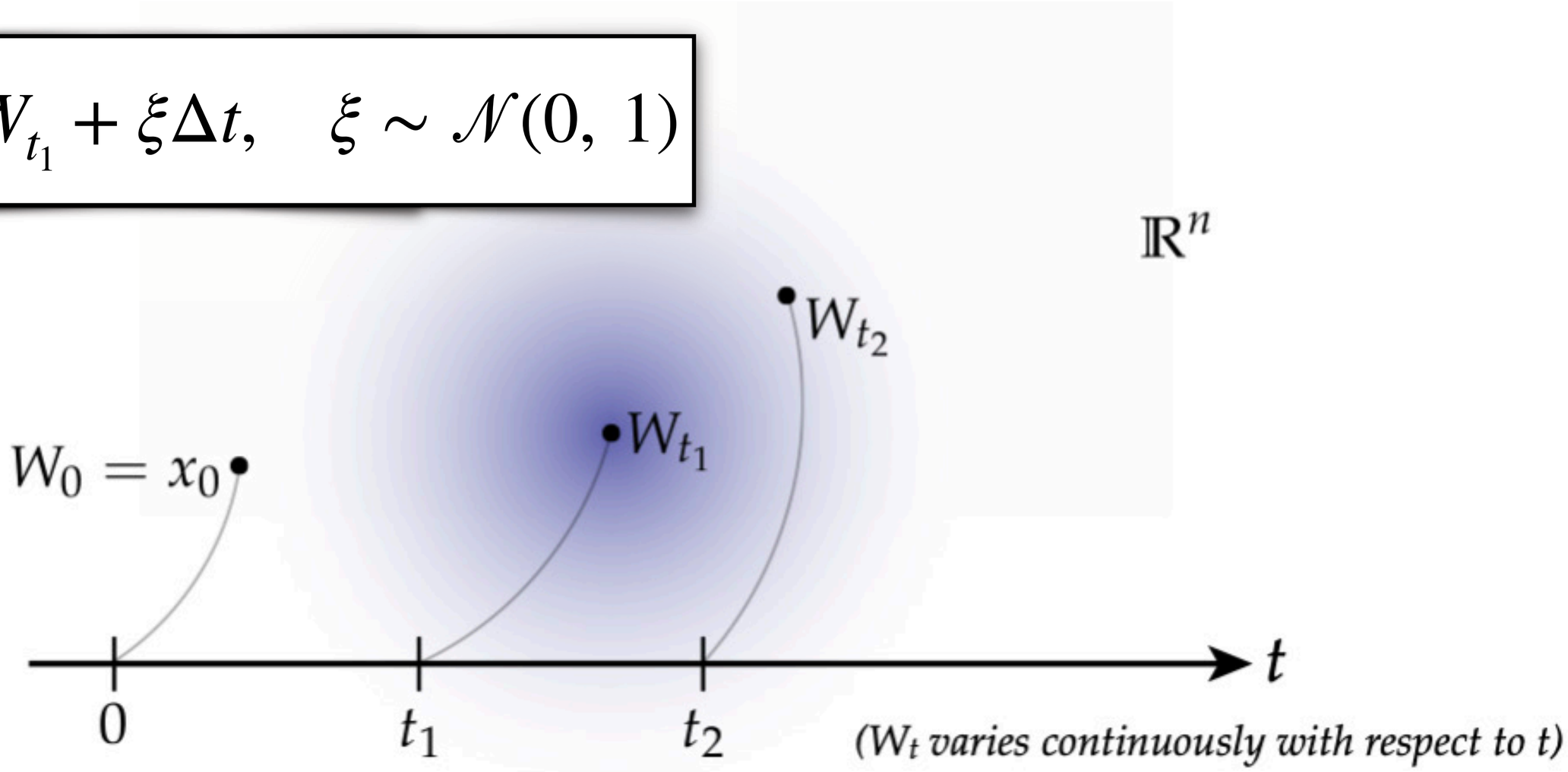


diffusion models in machine learning [Heitz et al, SIGGRAPH 2023]

Simulation of Brownian motion via Euler Muruyama

Collection of **independent normally-distributed increments**

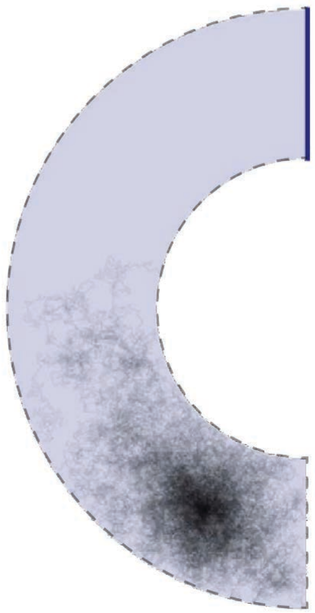
$$W_{t_2} \approx W_{t_1} + \xi \Delta t, \quad \xi \sim \mathcal{N}(0,\ 1)$$



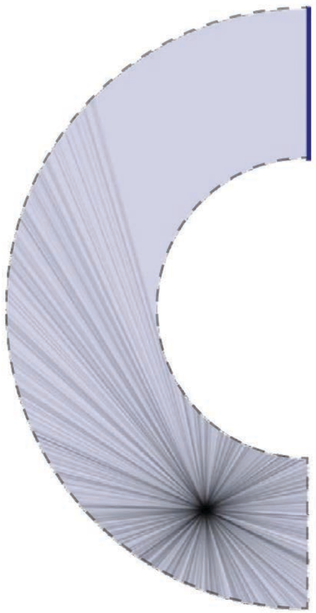
Numerical challenges in bounded domains

small time steps $\Delta t \rightarrow$ accurate results, long compute times

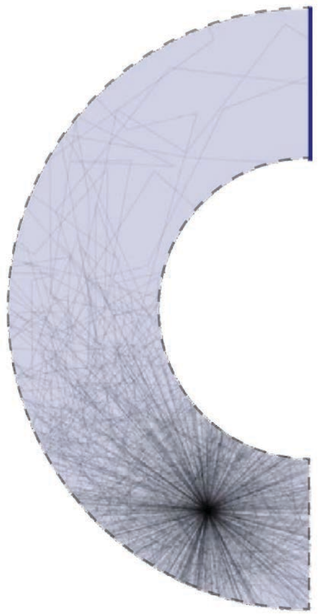
large time steps $\Delta t \rightarrow$ shorter compute times, large bias (error)



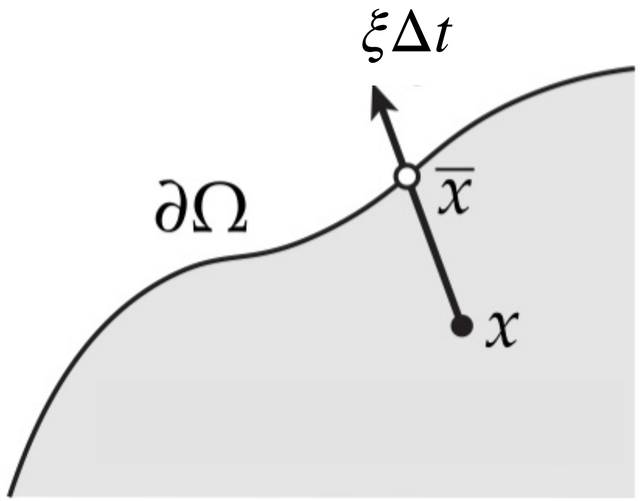
Δt too small



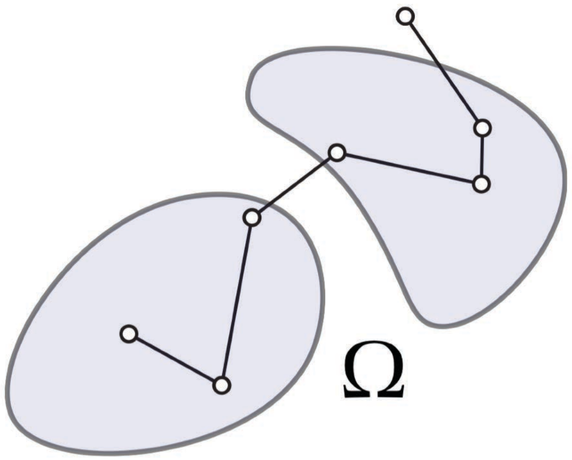
Δt too big



Δt just right?
(still biased)



truncate final step



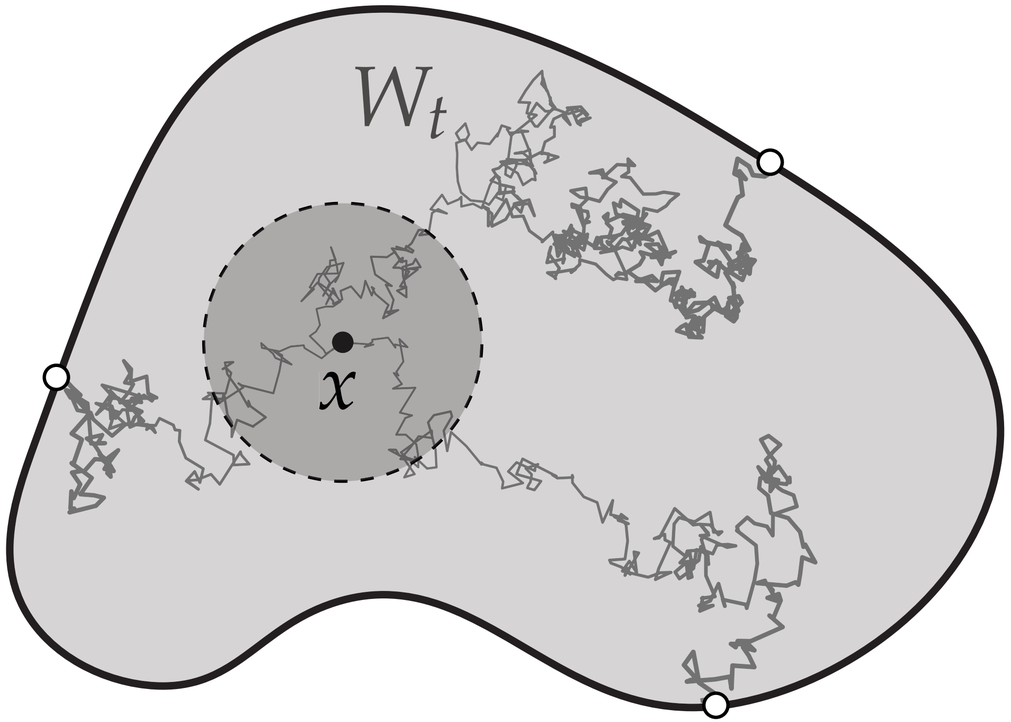
walks can jump
across domains

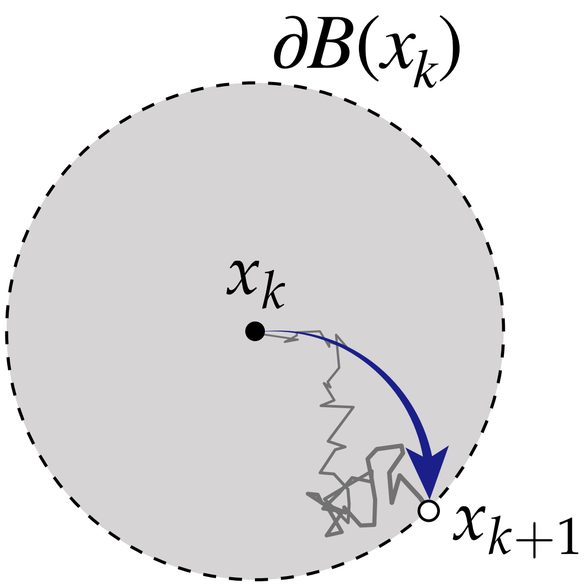
Simulation of Brownian motion via Walk on Spheres

Solution to Laplace eq can be computed by simulating **Brownian motion**

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

$$u(x) = \mathbb{E}[g(W_\tau)]$$
$$\approx \frac{1}{N}\sum_{i=1}^{N} g(W_\tau^i)$$





Brownian motion has a uniform exit distribution for a sphere of **any size**

Kakutani's Principle on a ball

$$\Delta u = 0 \text{ on } \Omega$$
$$u = g \text{ on } \partial\Omega$$

differential equation
(Laplace)

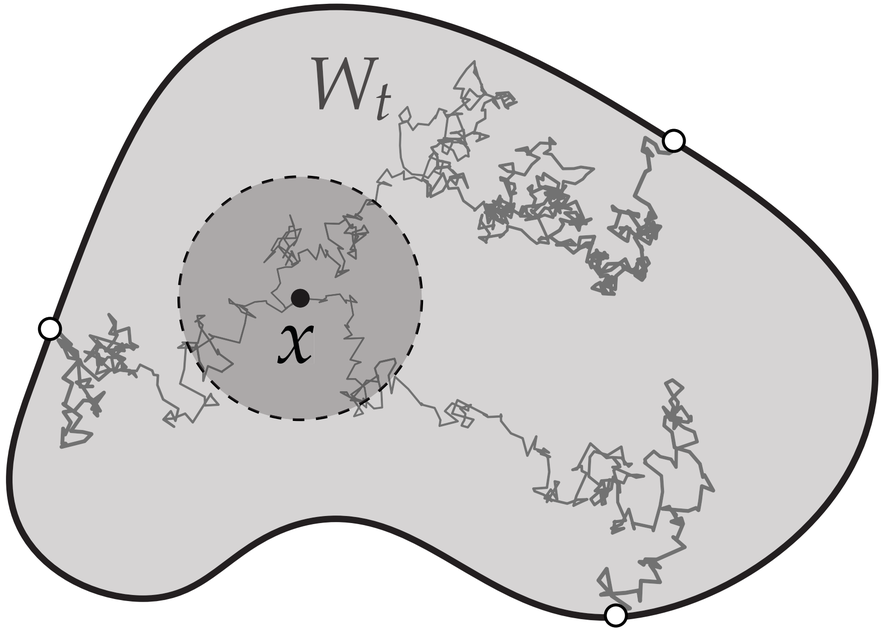
$$\mathbb{E}\left[g(W_T) \,\middle|\, W_0 = x \right]$$

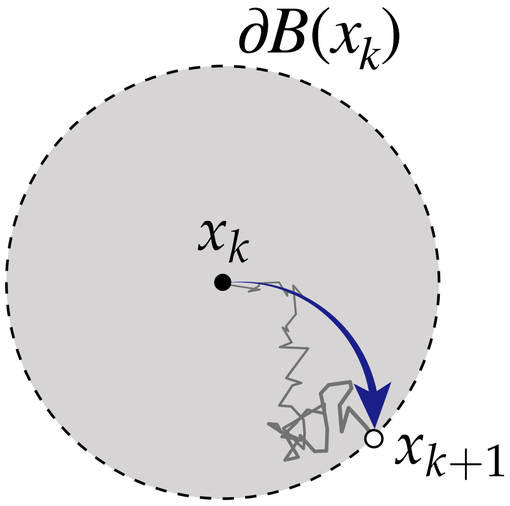
$T \rightarrow$ Time to reach boundary

Kakutani's Principle
(any domain)

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

mean value property
(on ball)





WoS provides
unbiased acceleration
of Brownian motion
in bounded domains!

Stochastic representation for source term

$$\Delta u = f \text{ on } \Omega$$
$$u = 0 \text{ on } \partial\Omega$$

differential equation
(Poisson)

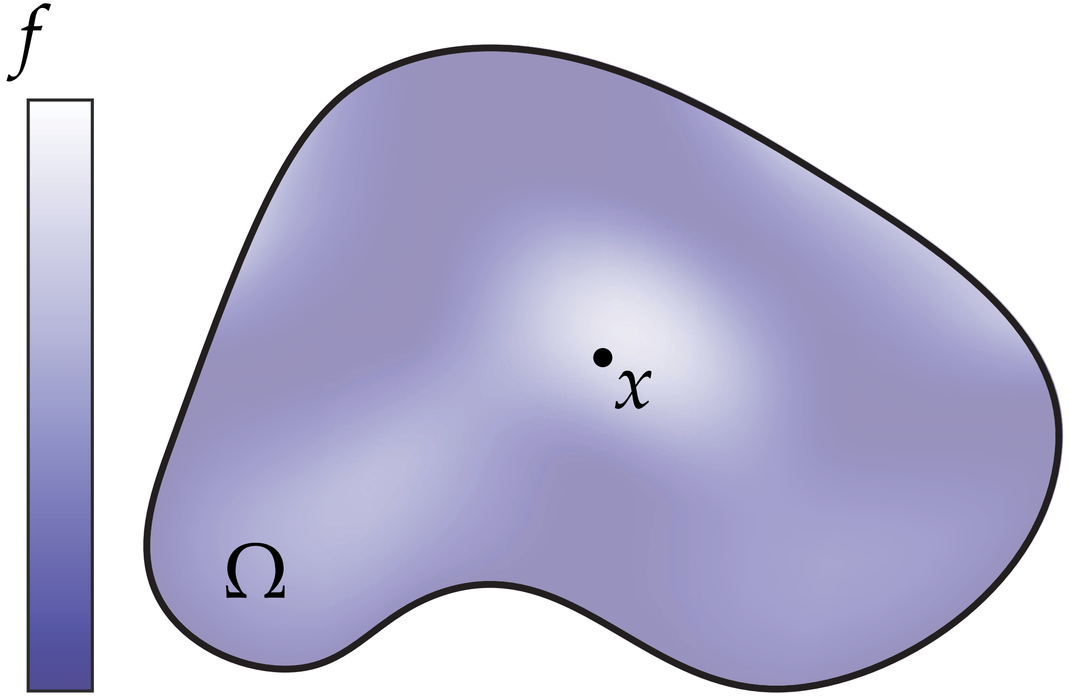
$$\mathbb{E}\left[\int_0^T f(W_t)\, dt \;\middle|\; W_0 = x\right]$$

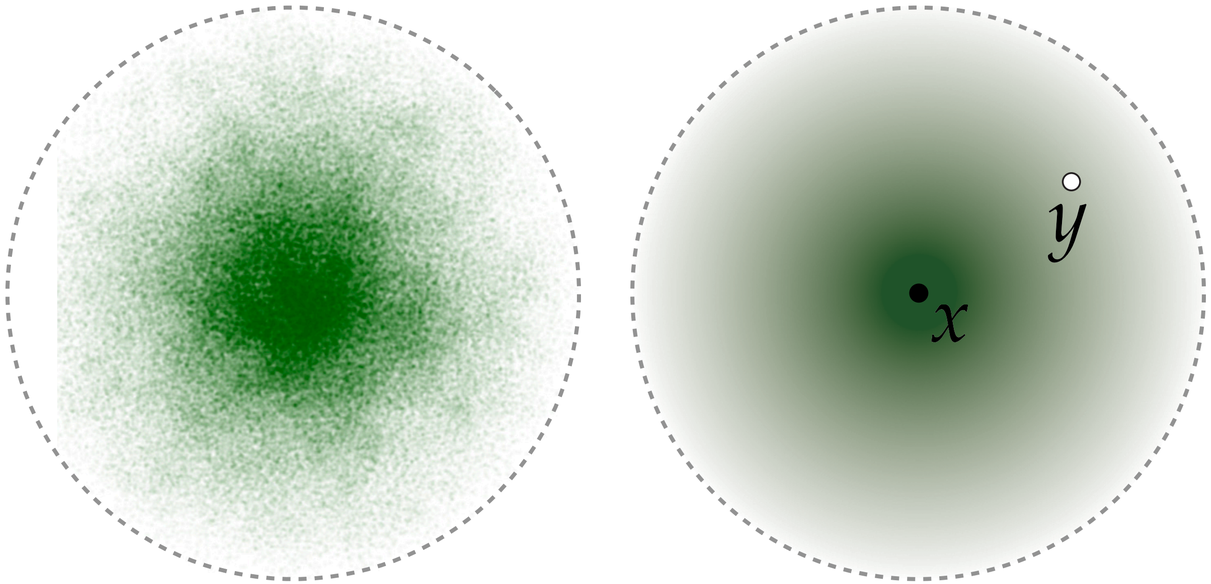
$T \to$ Time to reach boundary

stochastic representation
(any domain)

$$\int_{B(x)} f(y)\; G(x,y)\; dy$$

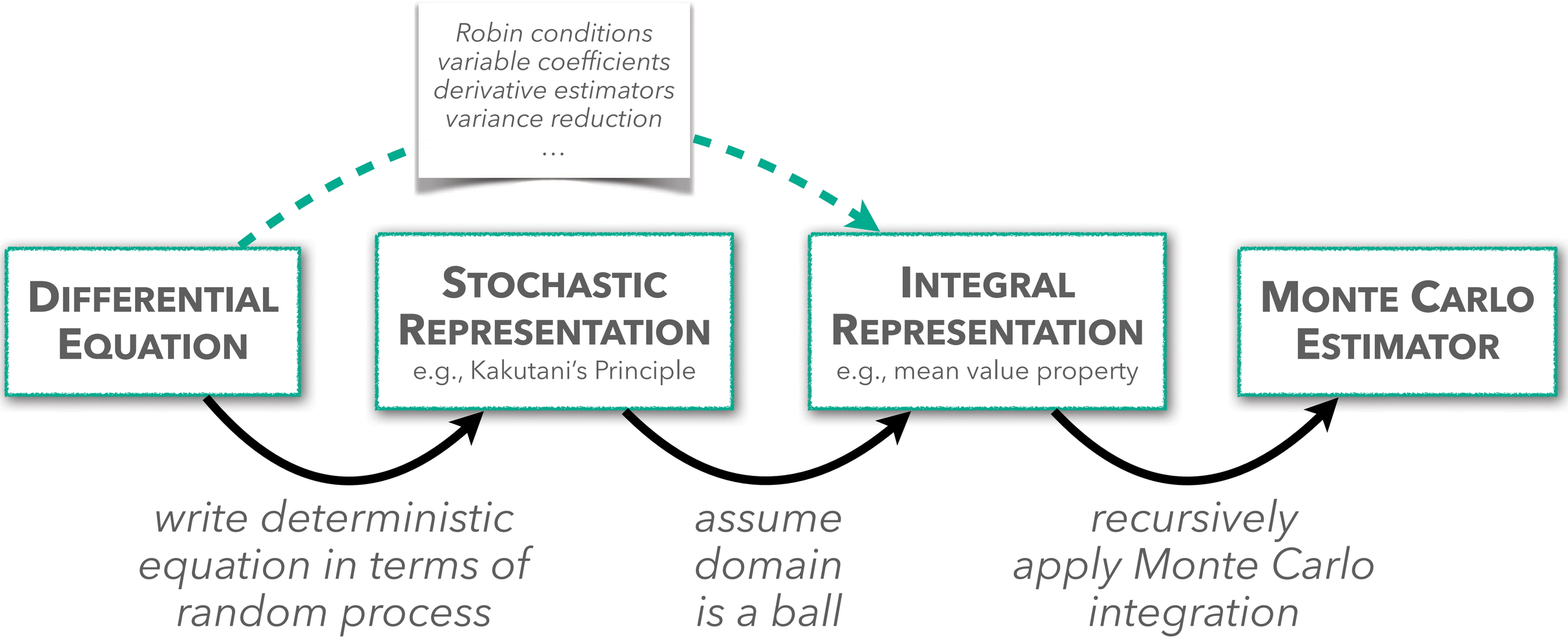
integral representation
(source term on ball)





Harmonic Green's Function $G(x, y)$

Deriving PDE estimators for Walk on Spheres

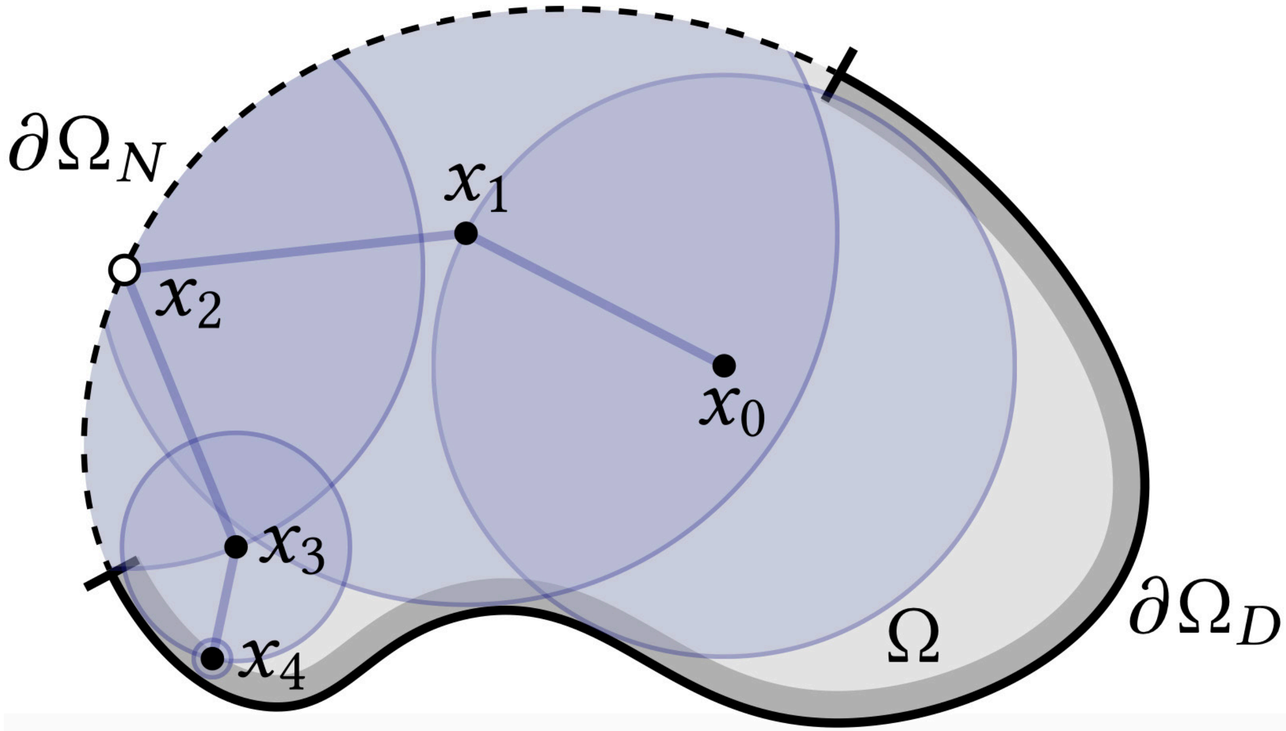


Walk on stars simulates Reflected Brownian motion

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

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

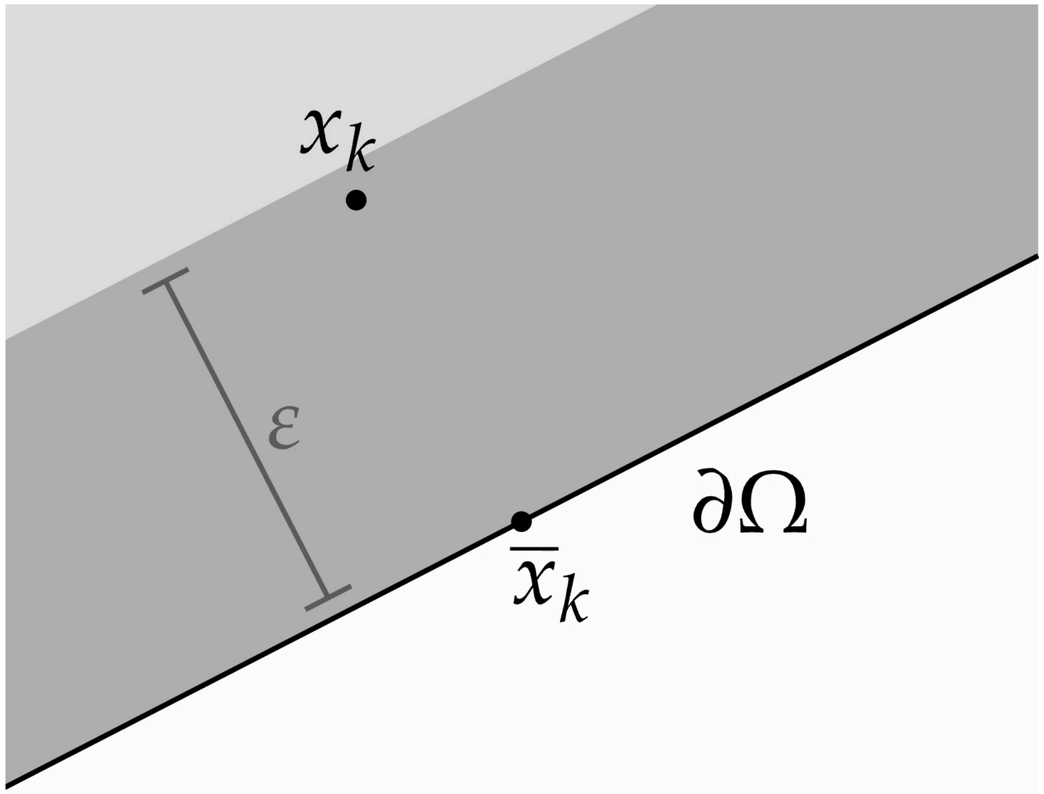
$$\frac{\partial u}{\partial n} = h \quad \text{on } \partial\Omega_N$$



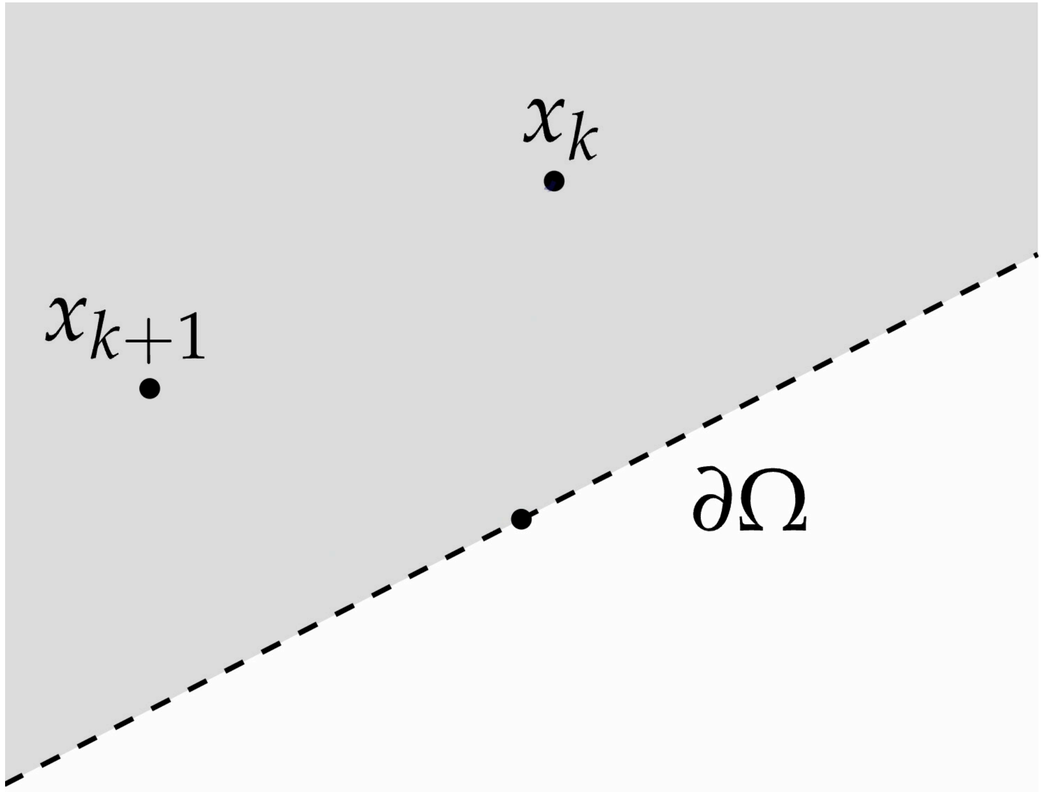
Walk on stars handles Neumann conditions by **reflecting random walks** on the boundary

Absorbed vs Reflected Brownian motion

Dirichlet ↔ absorbed



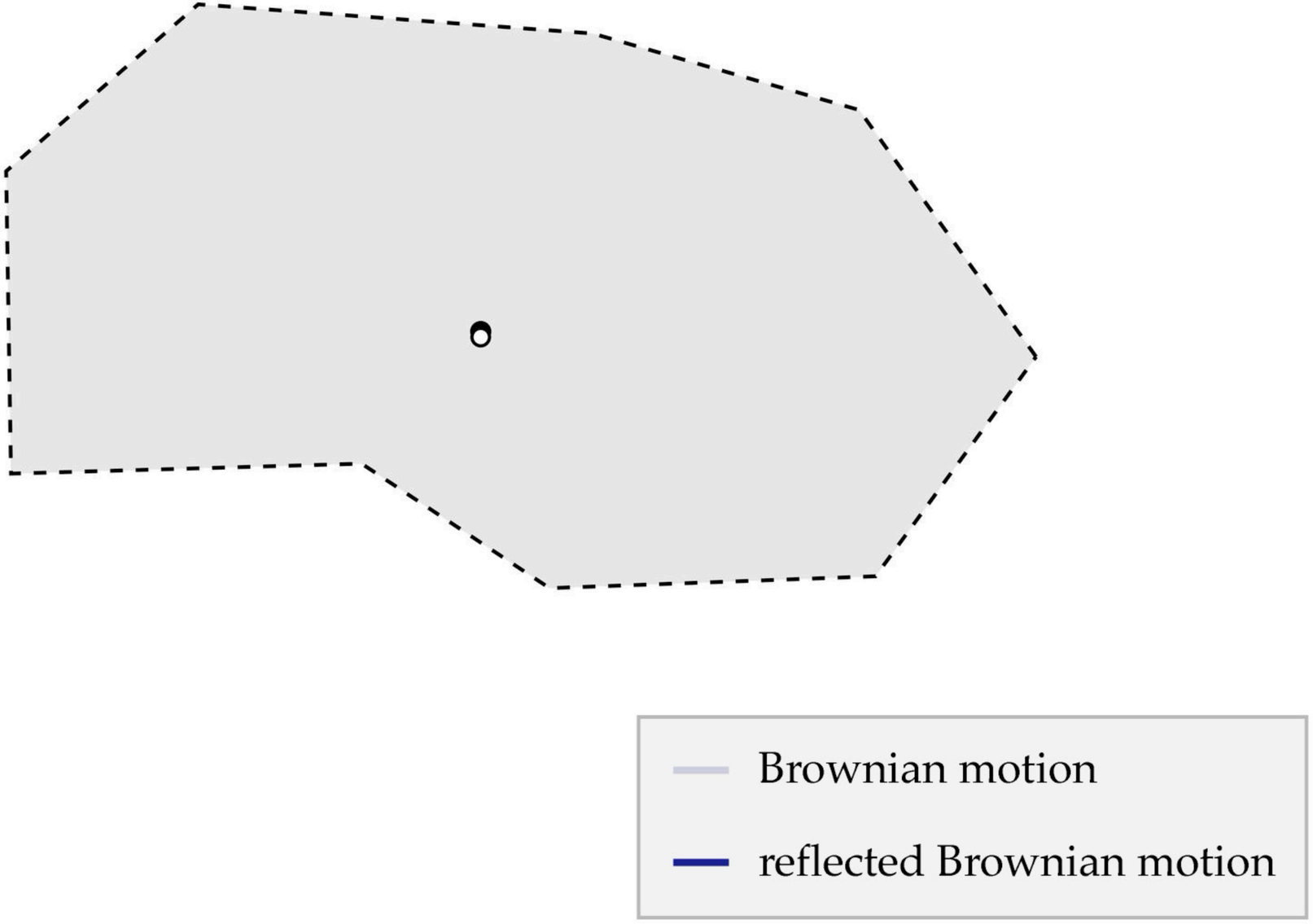
Neumann ↔ reflected



Reflected Brownian motion – 1D

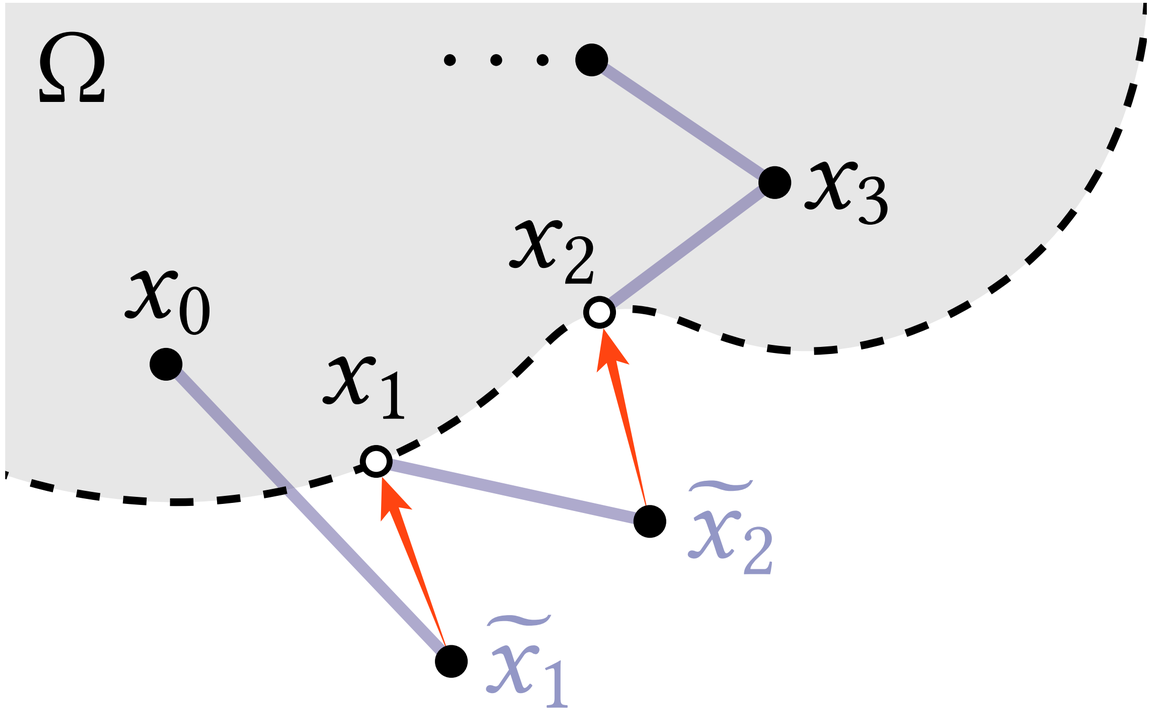
$$X_t := |W_t|$$

Reflected Brownian motion – nD / polyhedral



RBM simulation via Euler Muruyama

Numerical integration of Brownian motion is **slow** and **biased**



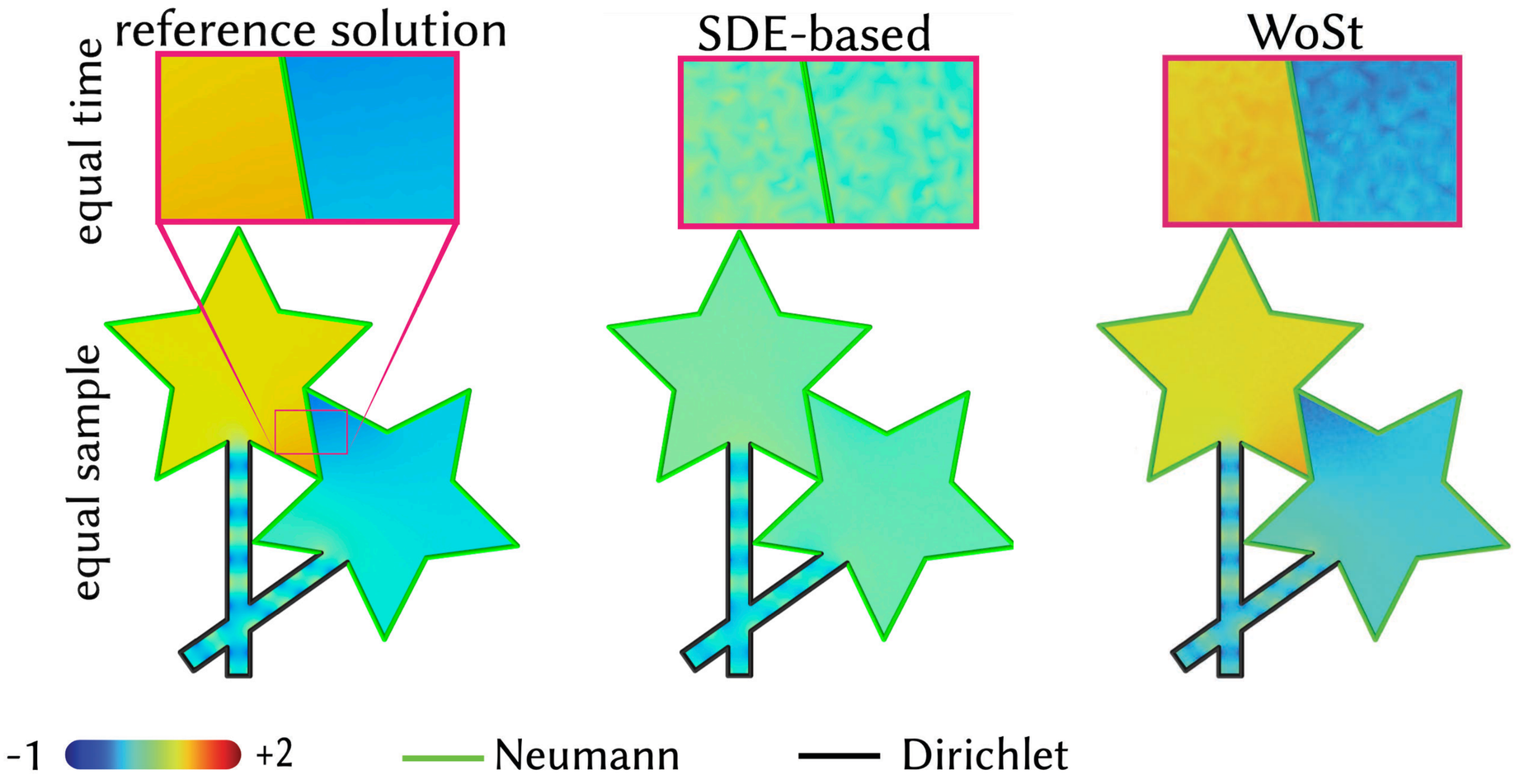
$$\xi \sim \mathcal{N}(0,\ 1)$$

$$W_{t_2} \approx W_{t_1} + \xi \Delta t$$

$$W_{t_2} \leftarrow \text{proj}_{\partial\Omega}(W_{t_2})$$

Comparison to alternatives: SDE integration

Numerical integration of Brownian motion is **slow** and **biased**



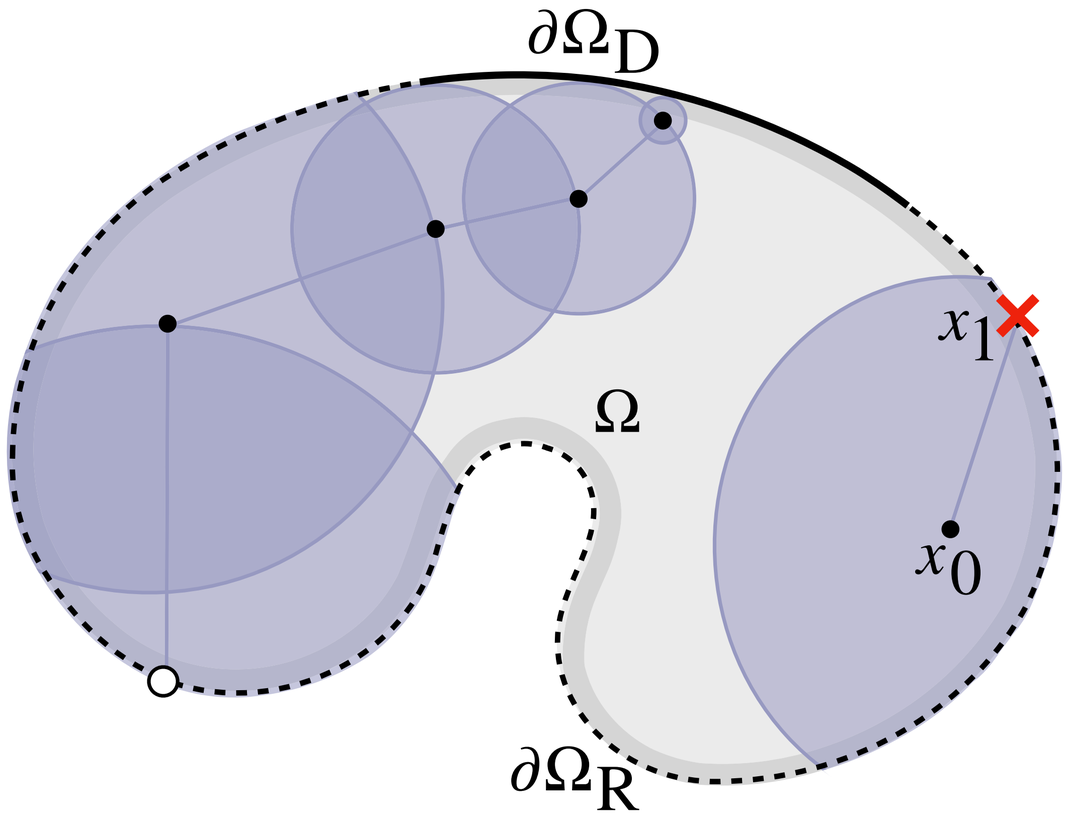
Robin boundary conditions

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

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

$$\frac{\partial u}{\partial n} = h \quad \text{on } \partial\Omega_N$$

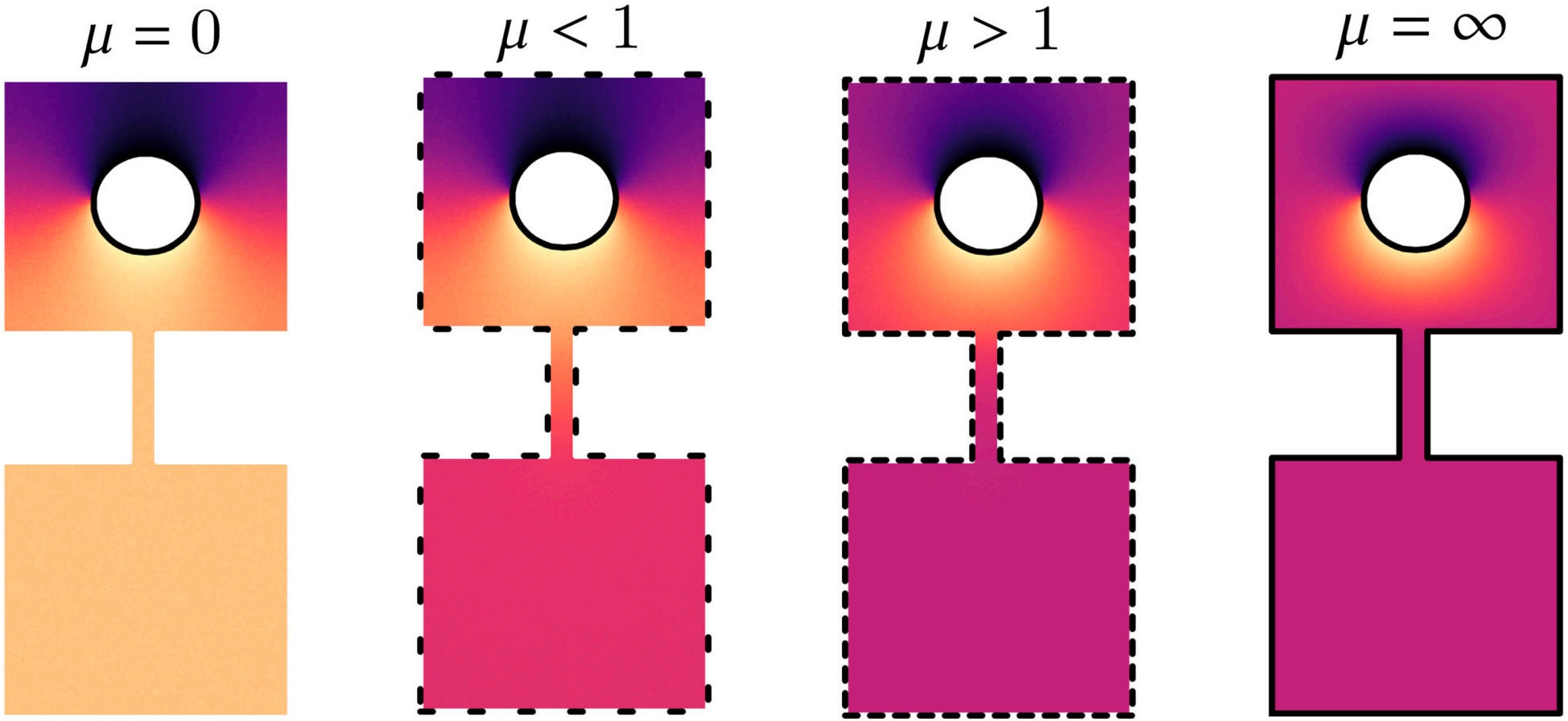
$$\frac{\partial u}{\partial n} - \mu u = k \quad \text{on } \partial\Omega_R$$



Walk on stars also handles Robin conditions
[Miller*, Sawhney*, Crane†, Gkioulekas†, 2024]

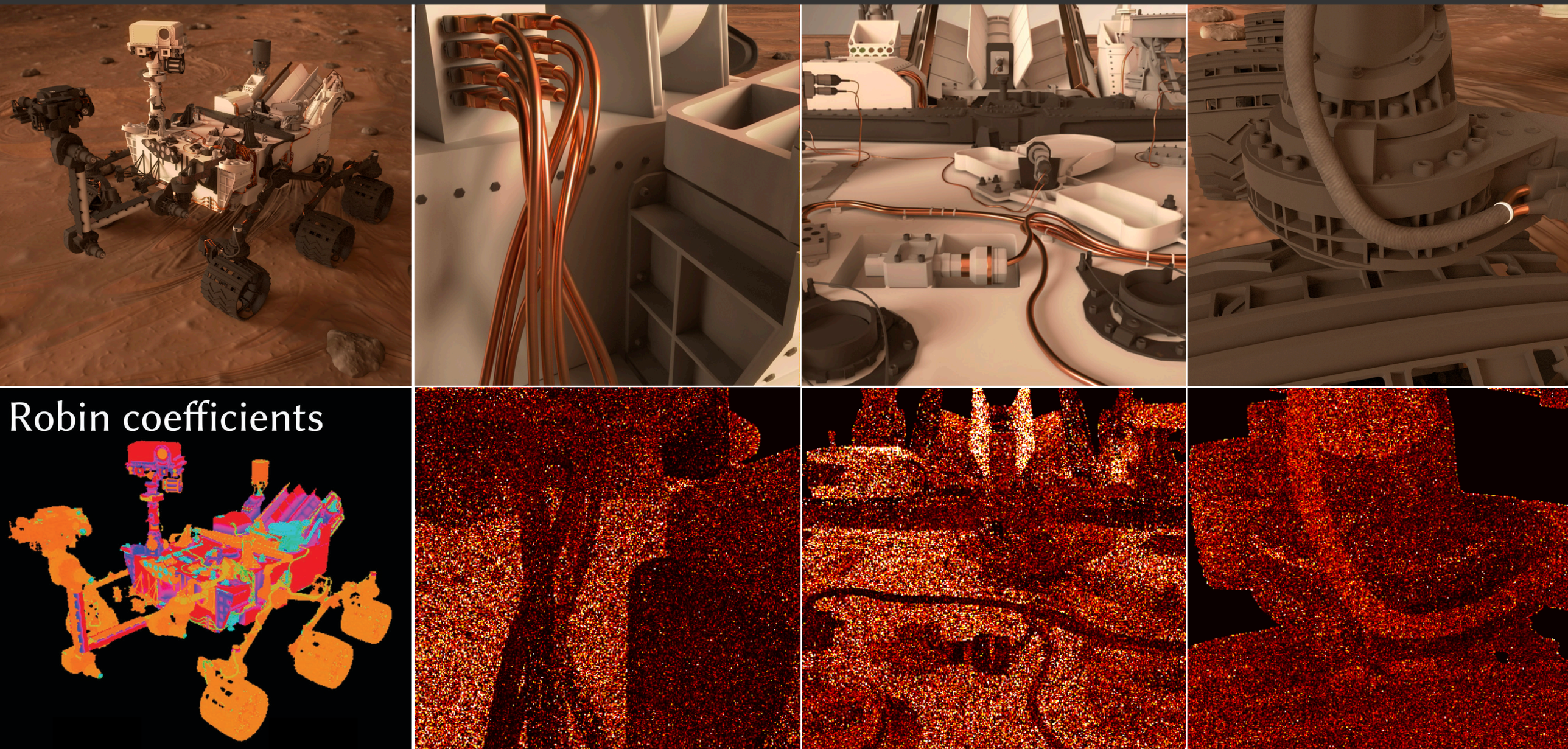
Robin boundary conditions

Linearly interpolate between Neumann and Dirichlet conditions



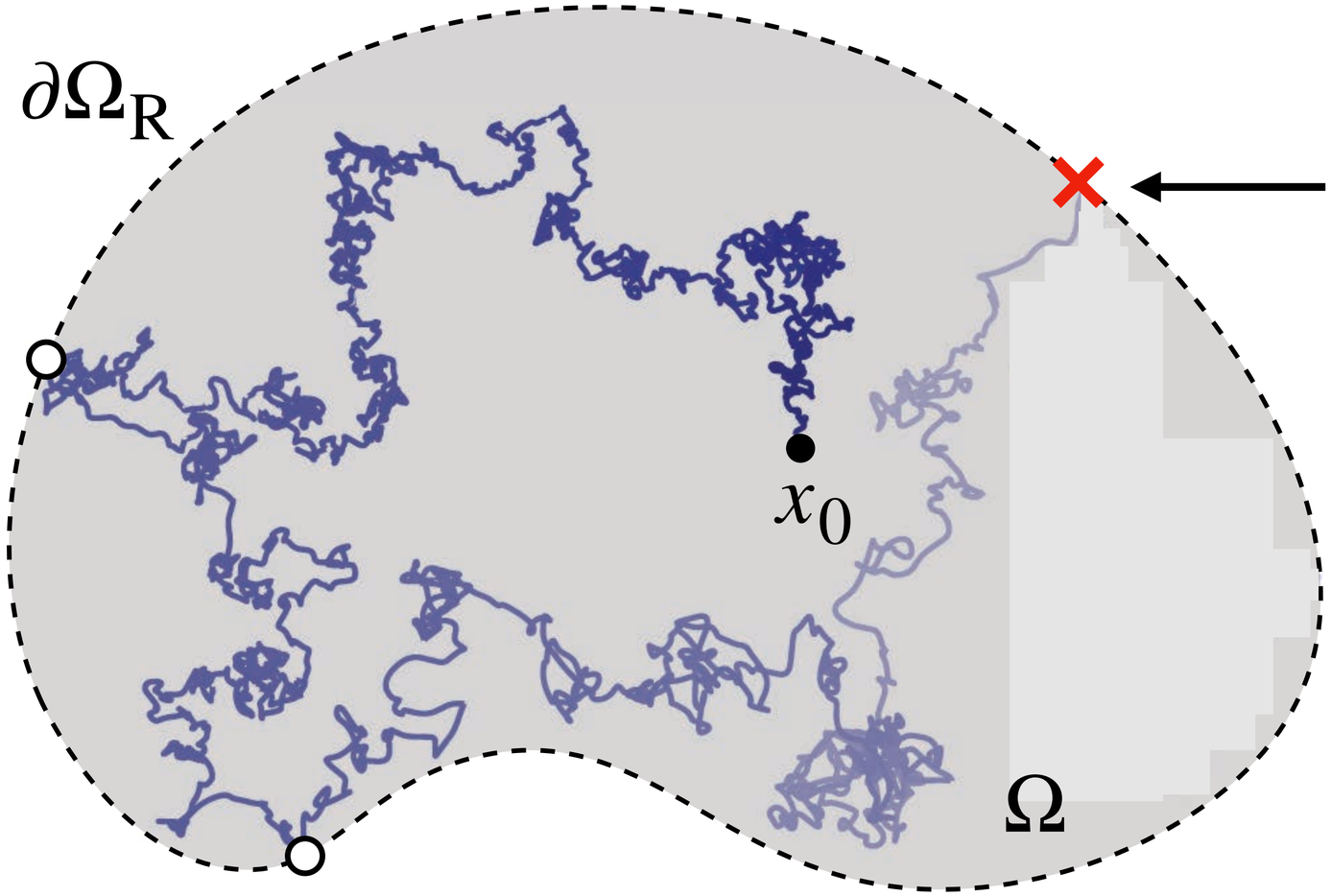
Neumann (purely reflecting) | **Robin** (more reflecting) | **Robin** (more absorbing) | **Dirichlet** (purely absorbing)

More accurate physical model



Partially reflected Brownian motion

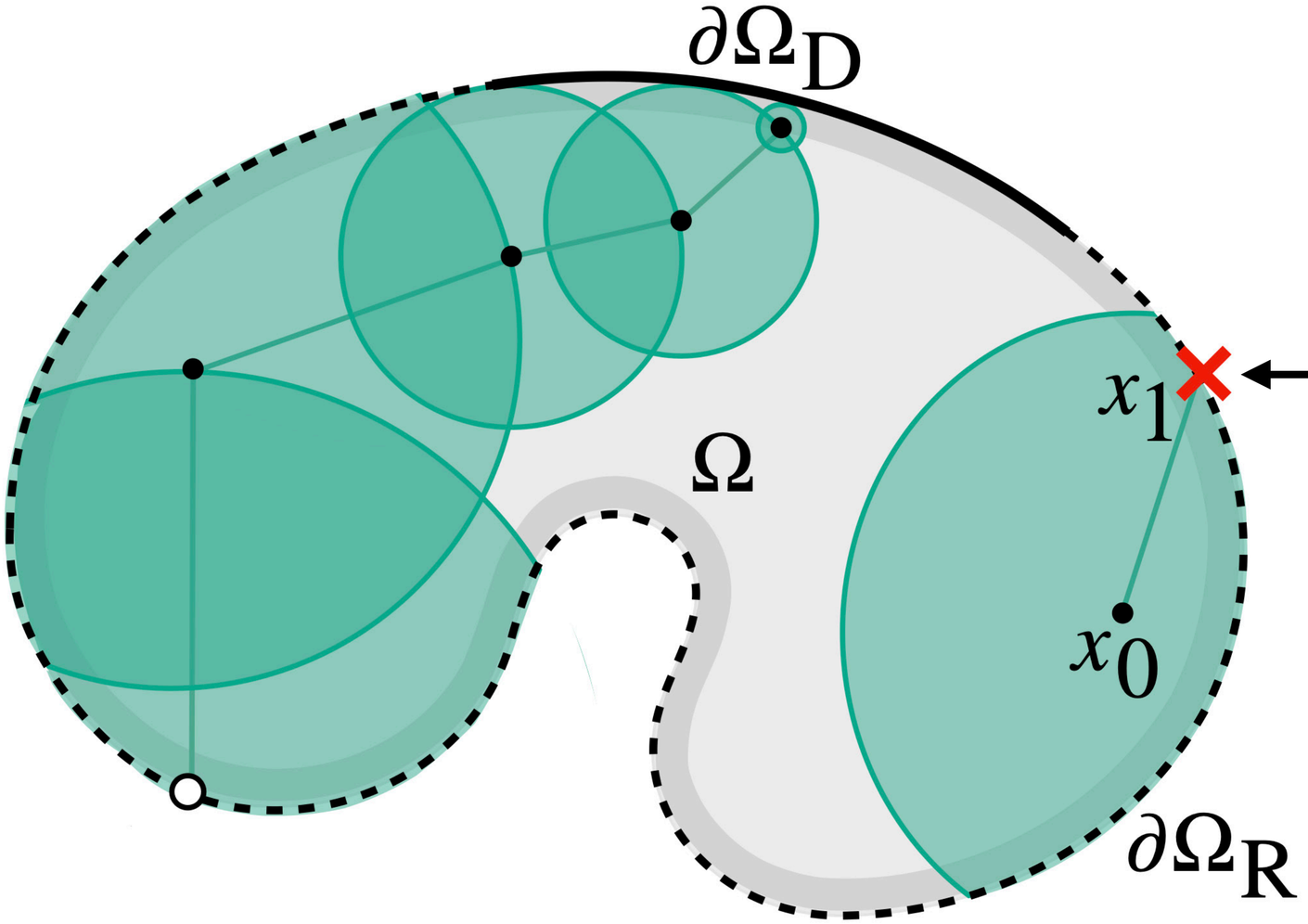
Non-zero probability of absorption on the boundary



random walk absorbed on Robin boundary

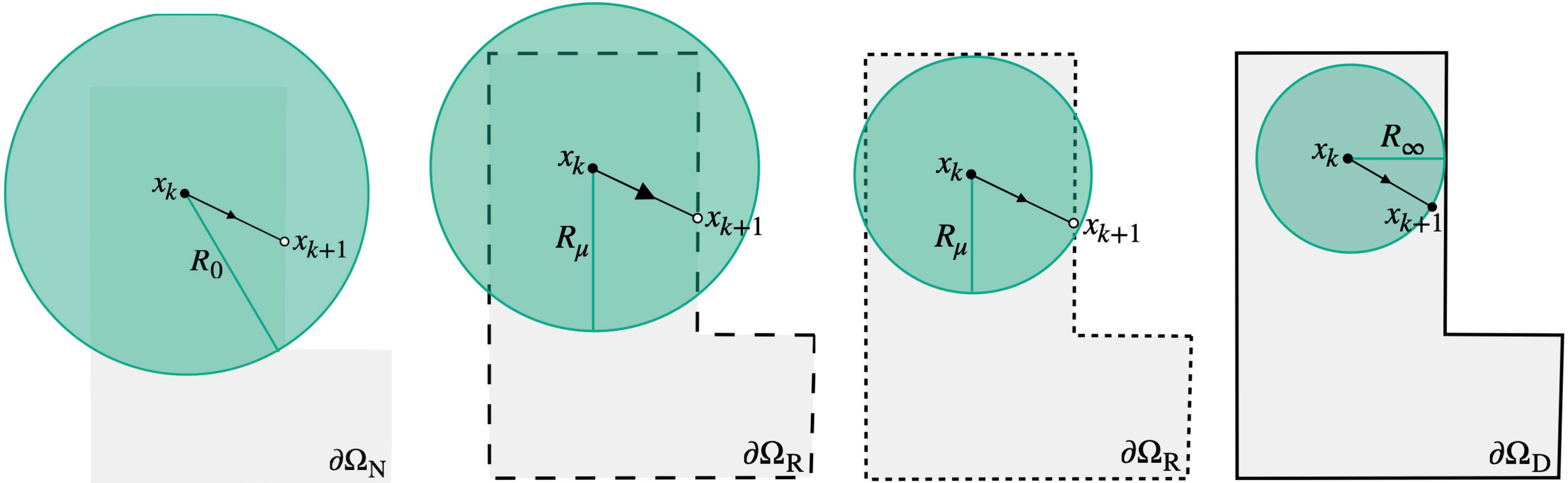
Walk on stars with Robin boundary conditions

Non-zero probability of absorption on the boundary



random walk terminated on Robin boundary

Radius of a star-shaped region with Robin conditions



Neumann
(purely reflecting)

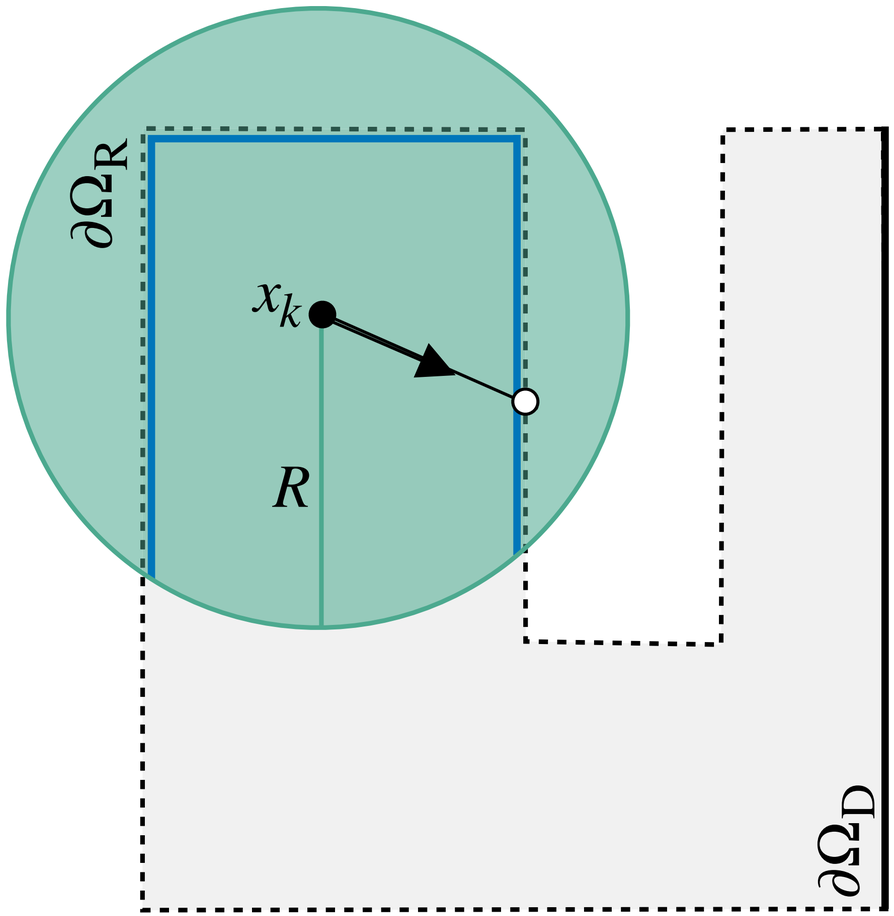
Robin
(more reflecting)

Robin
(more absorbing)

Dirichlet
(purely absorbing)

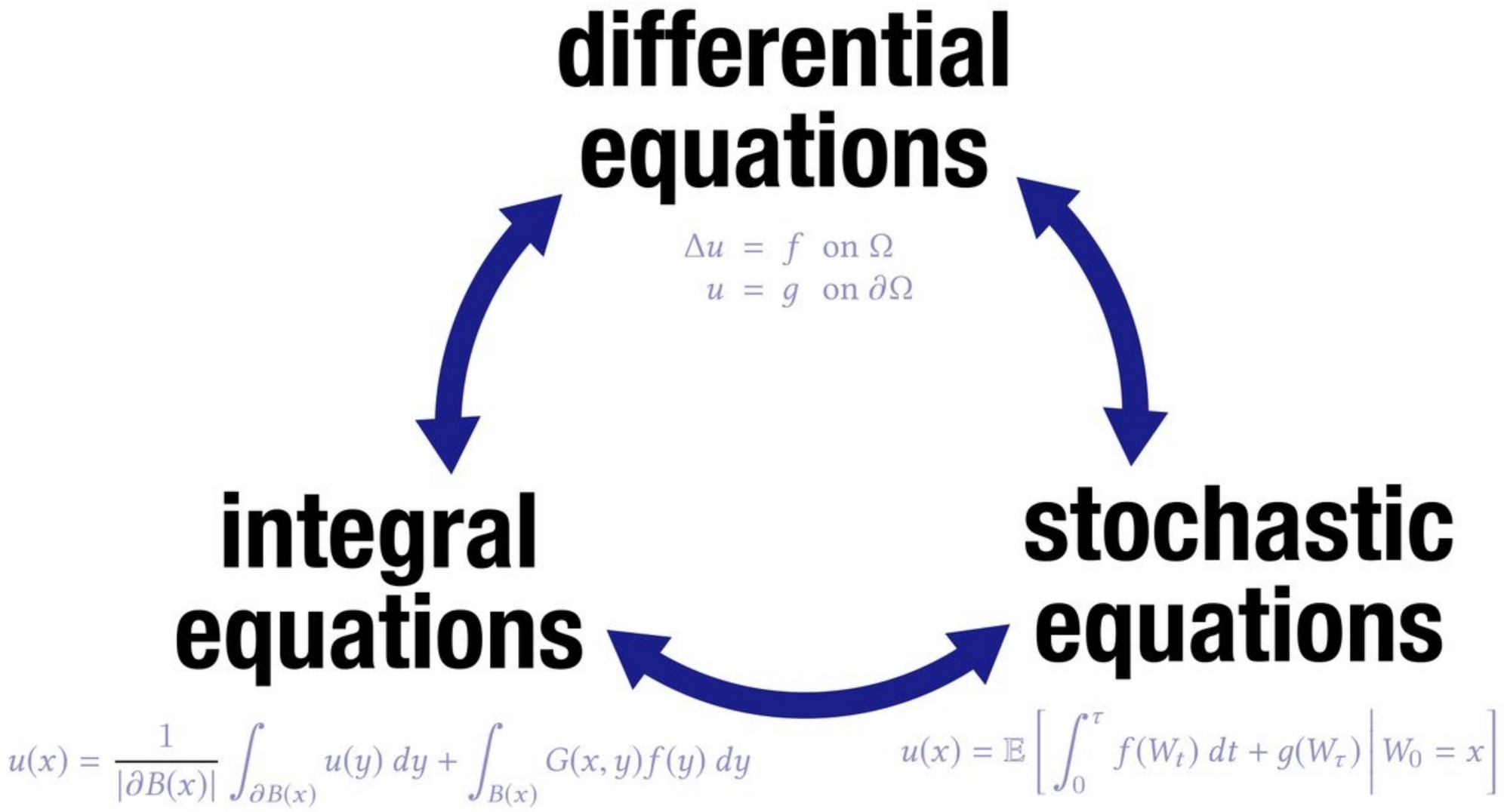
Reflectance function on Robin boundary

Reflectance ρ_μ is bounded between 0 and 1 with correct choice of radius

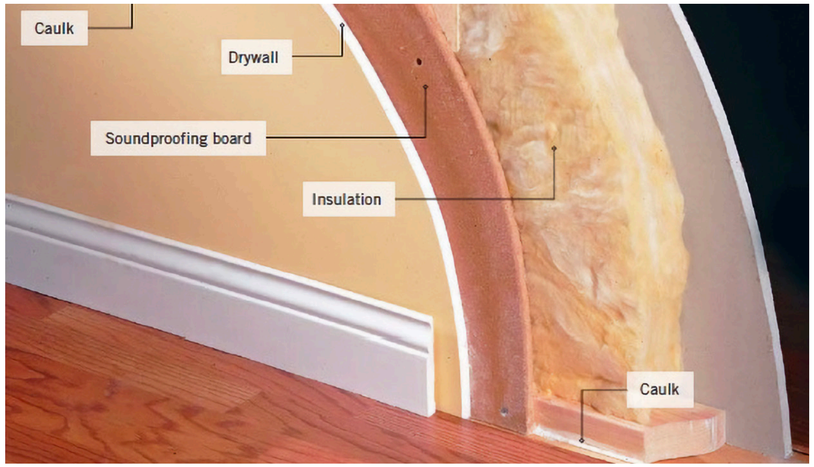


$$\rho_\mu \in [0,\ 1]$$

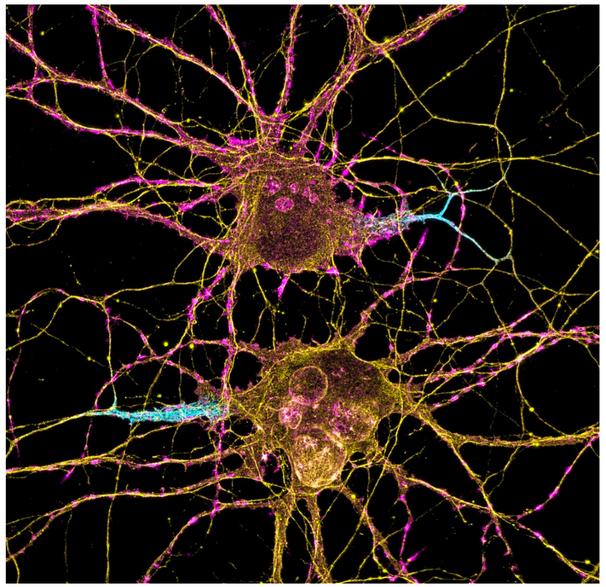
A tale of three types of equations…



Real systems exhibit spatial variation

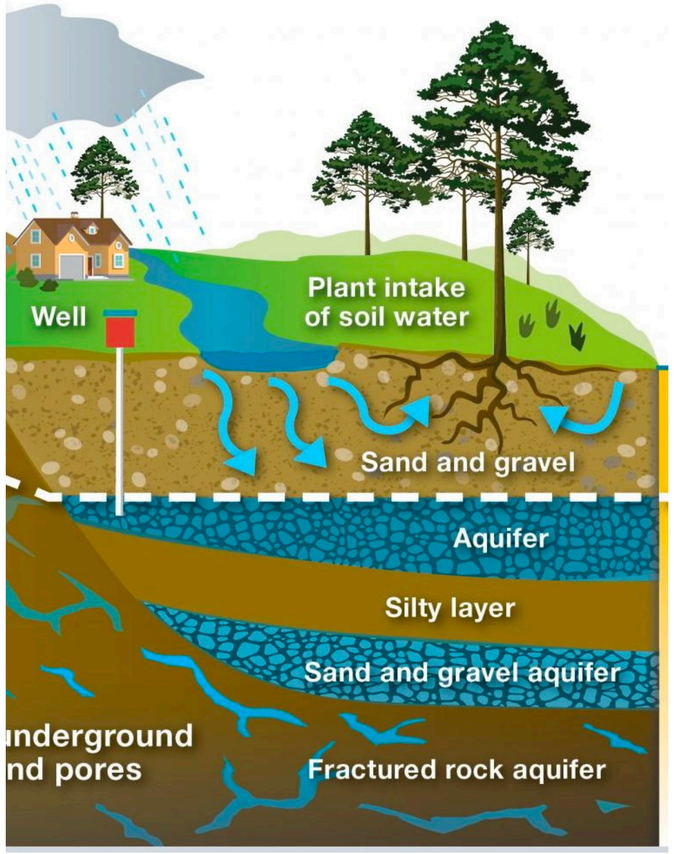


varying thermal diffusivity



varying electrical conductivity

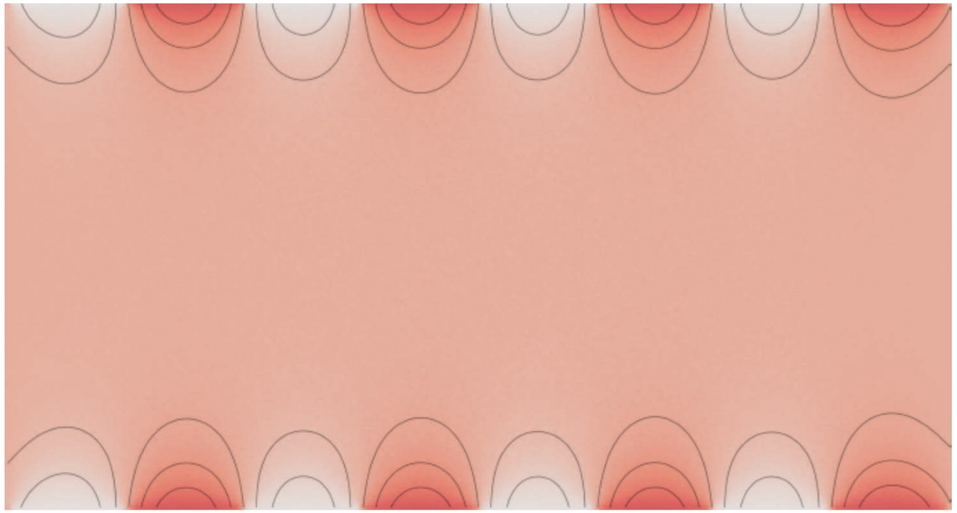
varying elastic response



varying permeability
of porous media

Laplace equation

boundary values $g(x)$



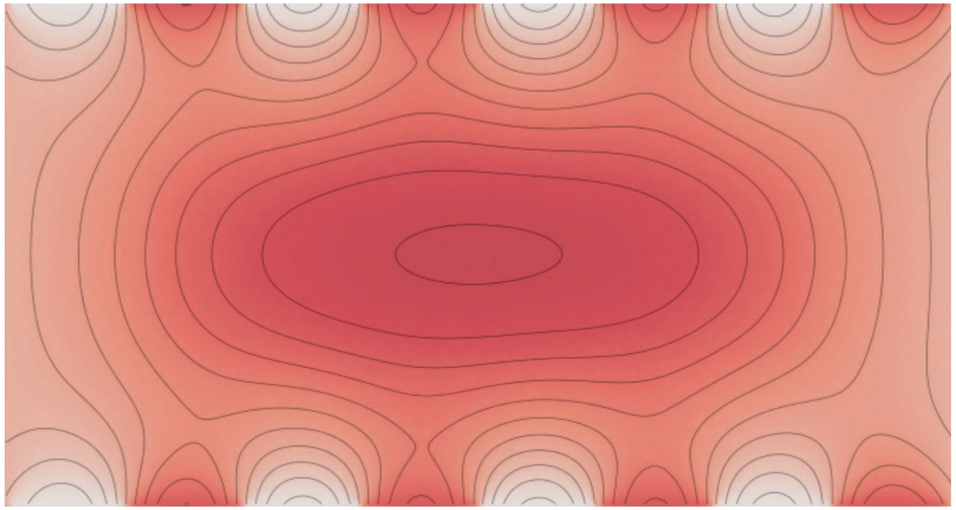
$$\Delta u = 0$$

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

Intuition: temperature along boundary is fixed

Poisson equation

source
term $f(x)$

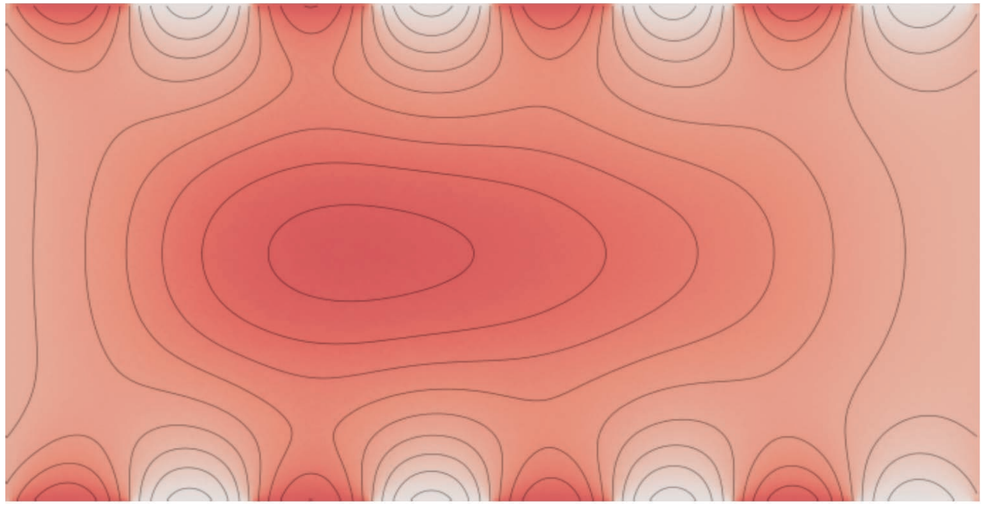


$$\Delta u = f$$

Intuition: adds additional “background temperature”

Variable diffusion Poisson equation

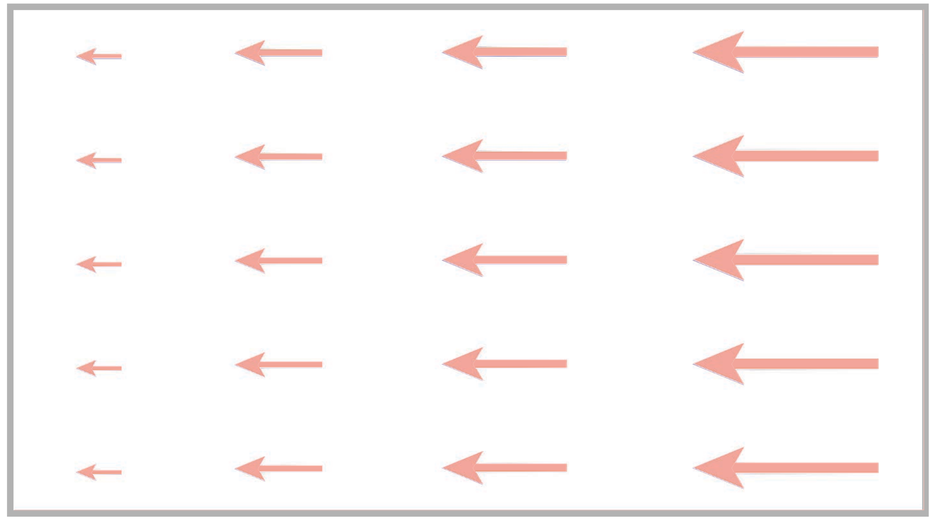
diffusion coefficient $\alpha(x)$



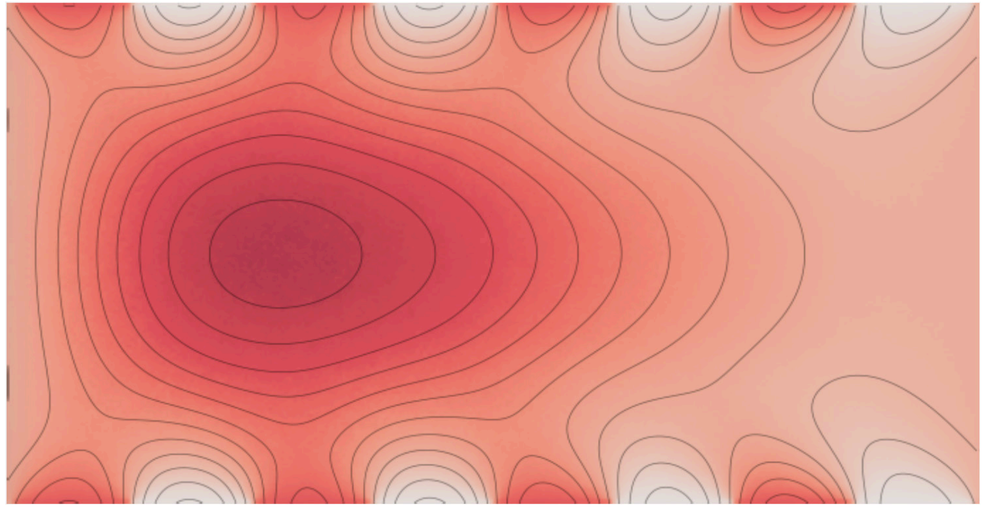
$$\nabla \cdot (\alpha \nabla u)\, u = f$$

Intuition: how fast does heat "spread out"?

Stationary advection-diffusion equation



transport coefficient $\overrightarrow{\omega}(x)$

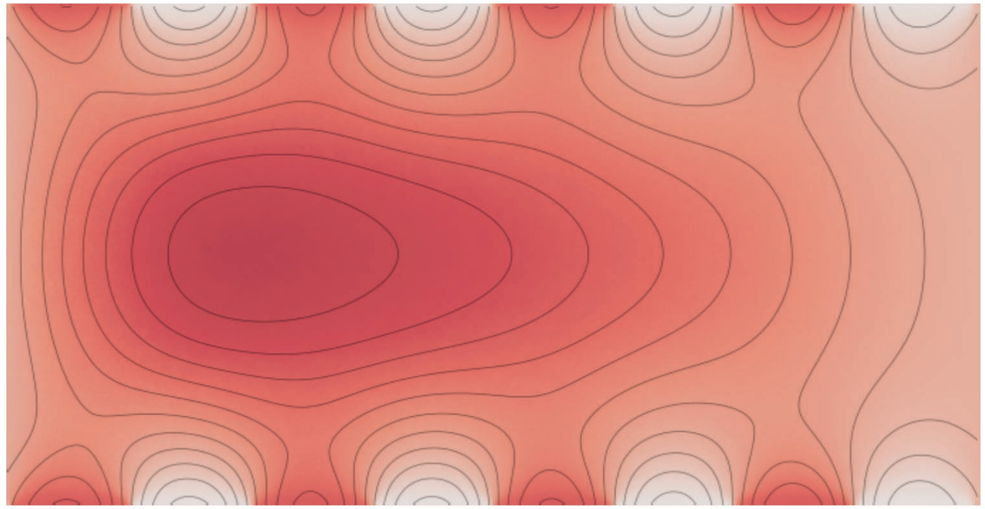


$$\Delta u + \overrightarrow{\omega} \cdot \nabla u = f$$

Intuition: heat is dragged along with a flowing river

Screened Poisson equation

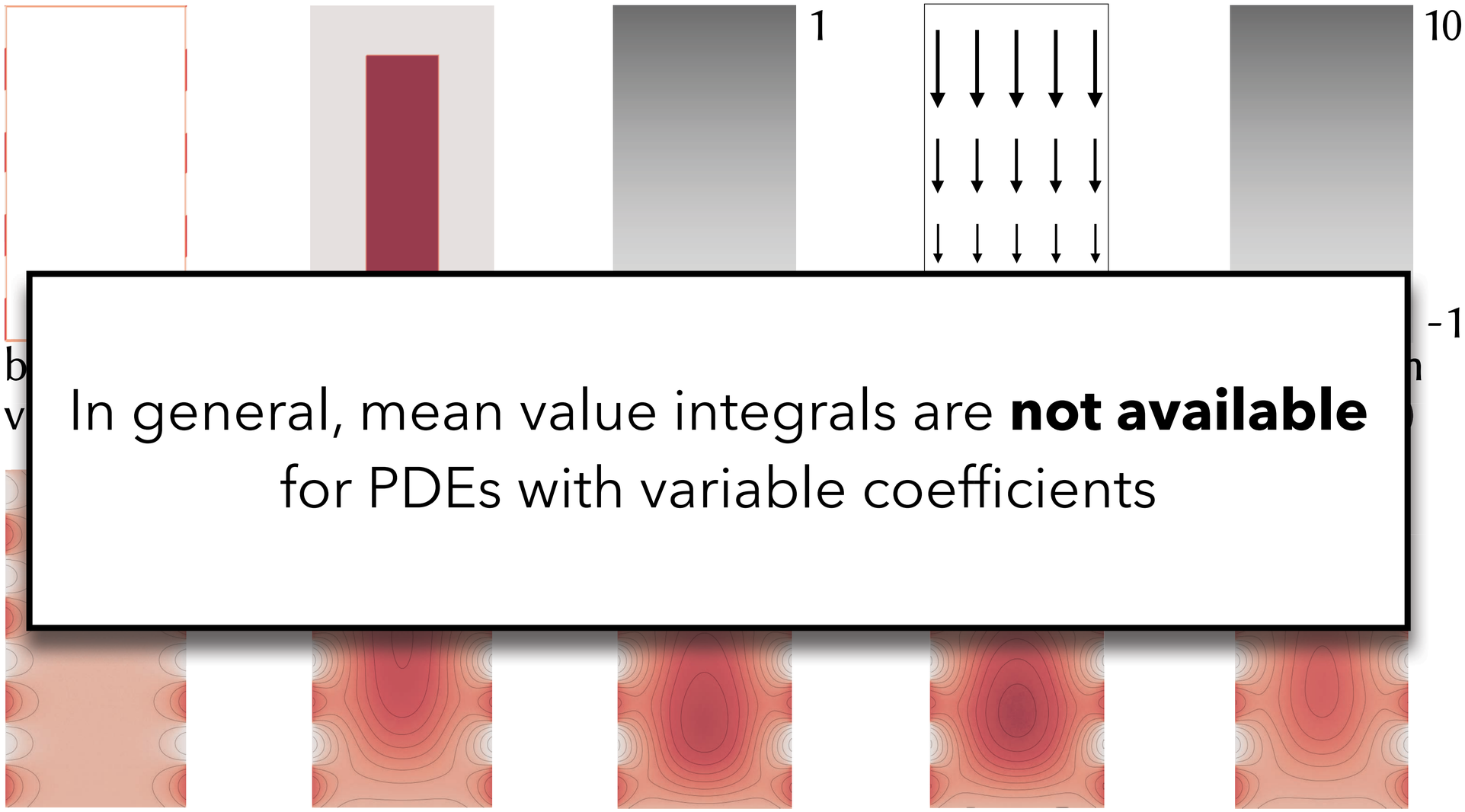
absorption coefficient $\sigma(x)$



$$\Delta u - \sigma u = f$$

Intuition: “cooling” due to absorption into background medium

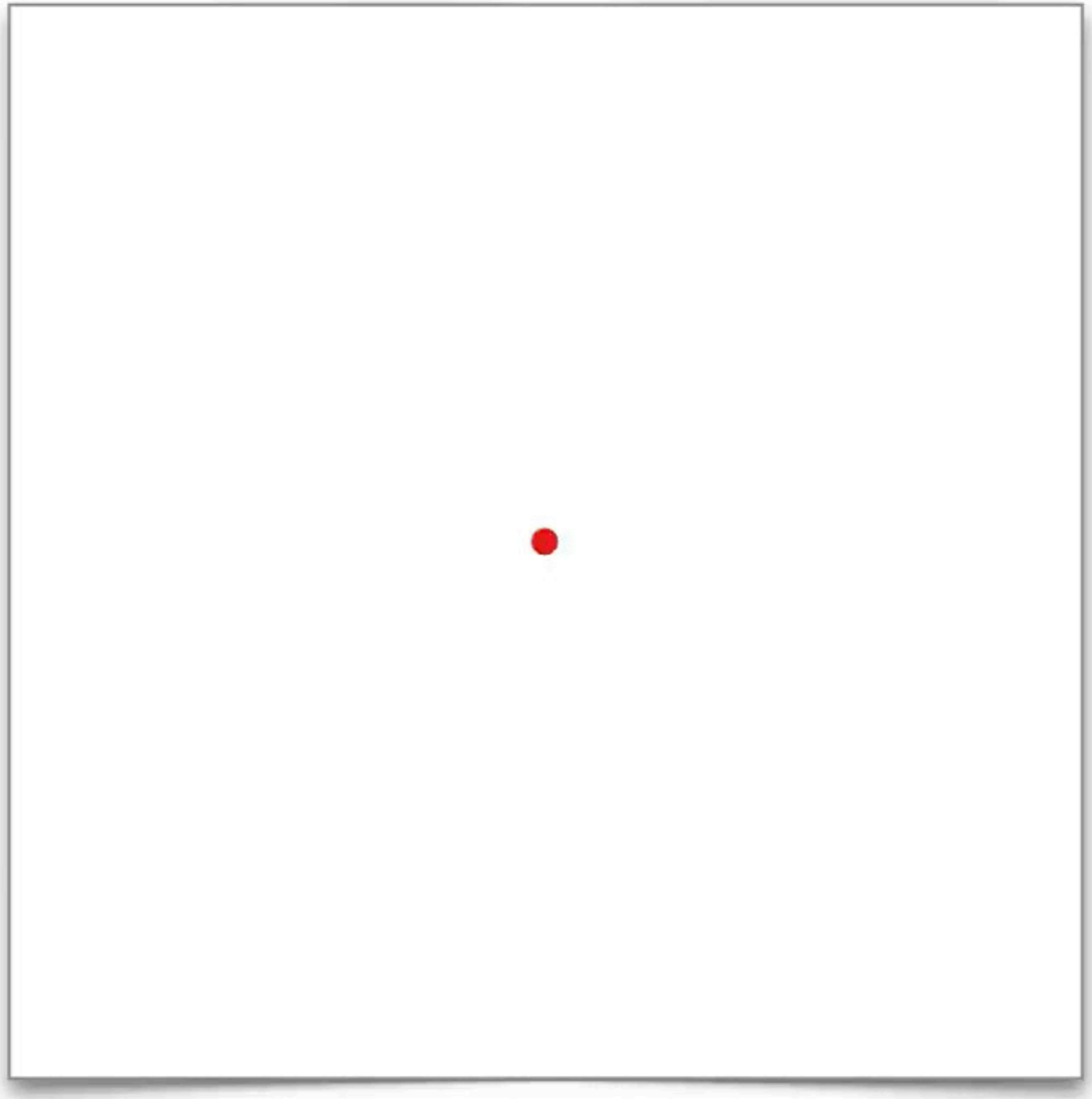
2nd order linear elliptic PDEs



In general, mean value integrals are **not available** for PDEs with variable coefficients

$\Delta u = 0$ $\quad$ $\Delta u = f$ $\quad$ $\nabla \cdot (\alpha \nabla u) = f$ $\quad$ $\Delta u + \vec{\omega} \cdot \nabla u = f$ $\quad$ $\Delta u - \sigma u = f$

Stochastic differential equations (SDEs)

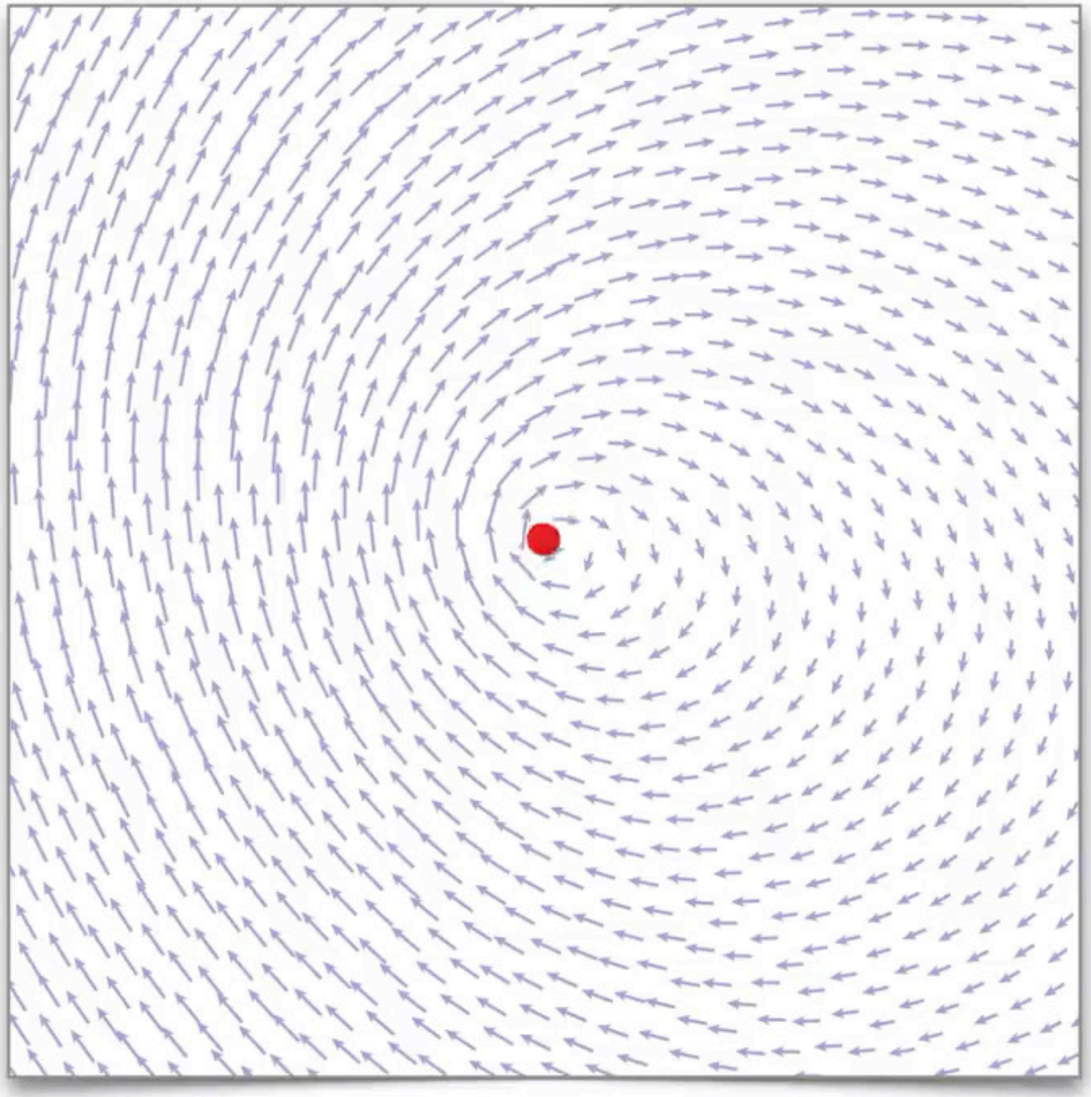


Brownian Motion

$$dX_t = dW_t$$

trajectory (X_t)

Stochastic differential equations (SDEs)



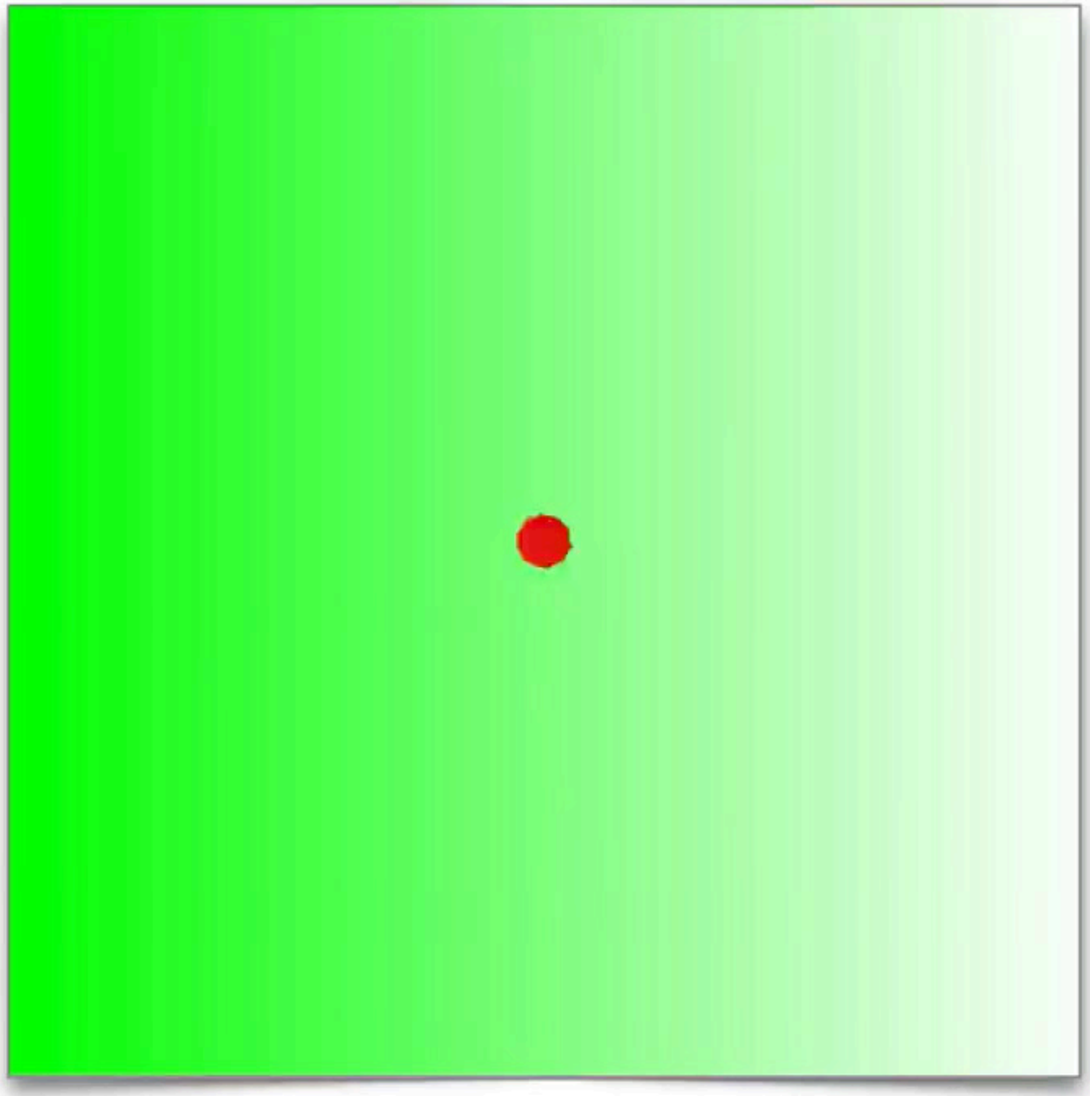
Brownian Motion with drift

$$dX_t = \vec{\omega}(X_t)\,dt + dW_t$$

trajectory (X_t)

drift direction $(\vec{\omega})$

Stochastic differential equations (SDEs)



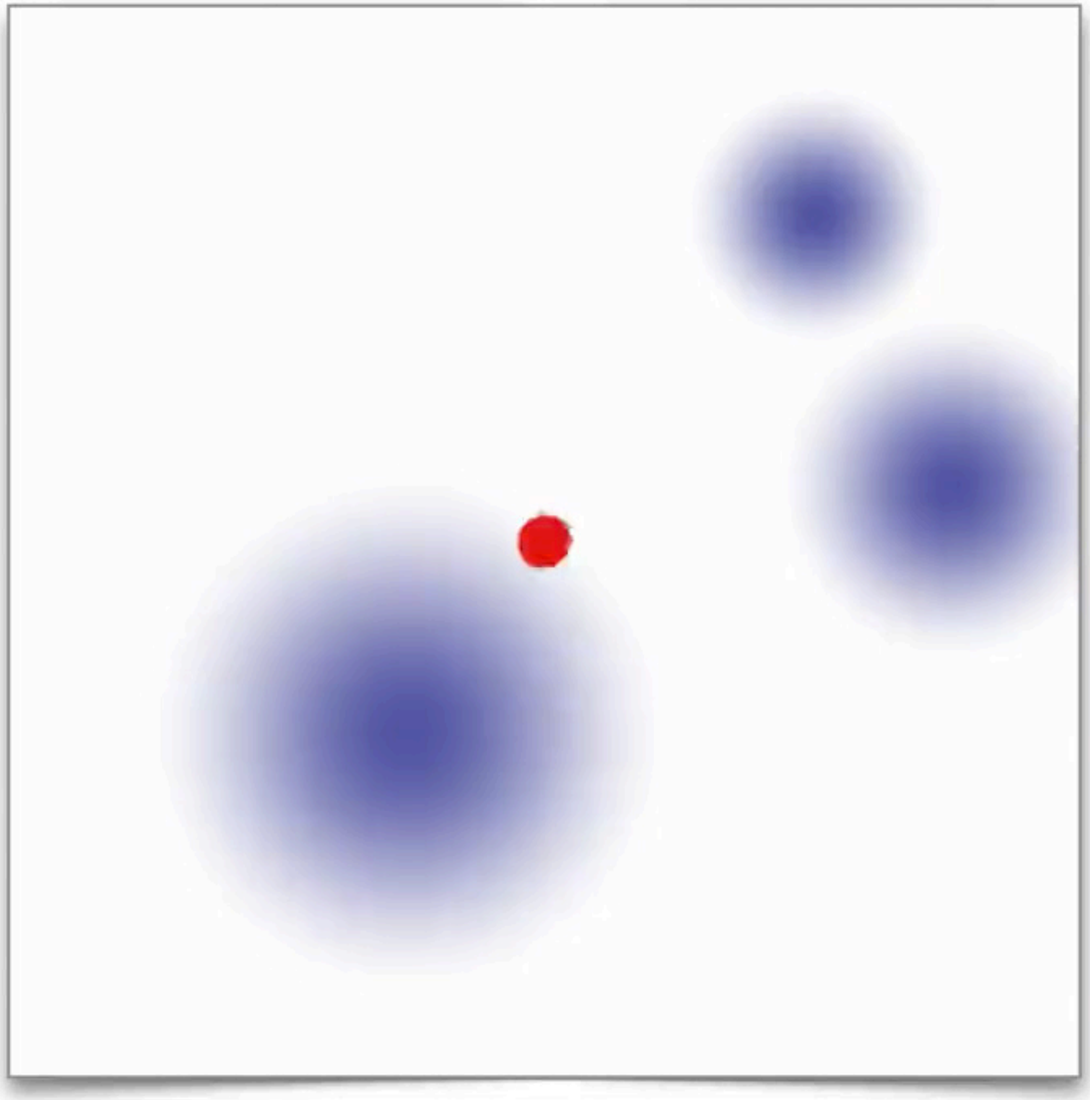
Brownian Motion with variable scale

$$dX_t = \sqrt{\alpha(X_t)}\, dW_t$$

trajectory (X_t)

diffusivity (α)

Stochastic differential equations (SDEs)



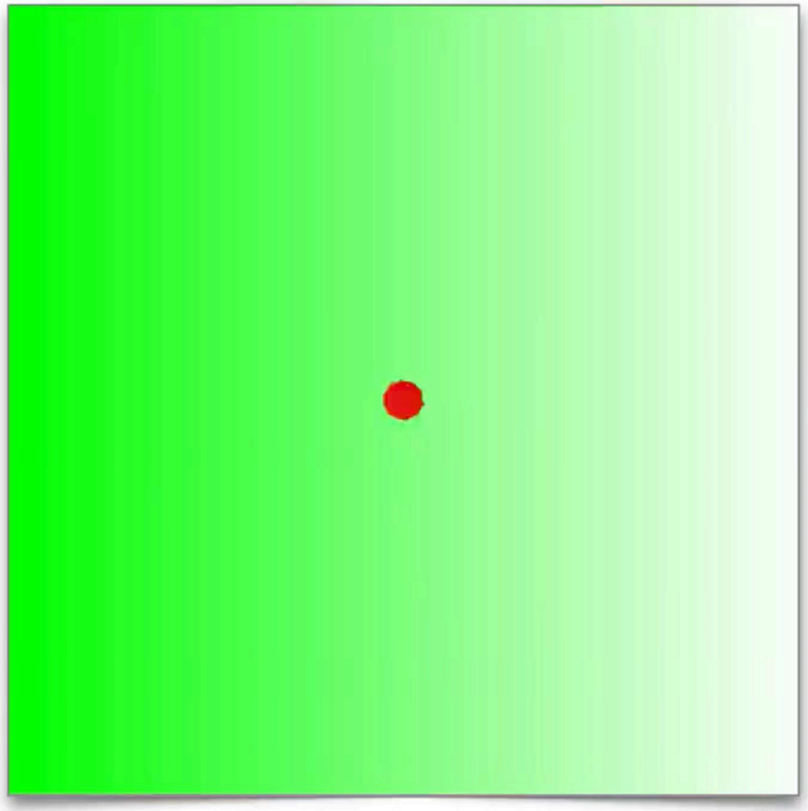
Brownian Motion in absorbing medium

$$dX_t = dW_t$$

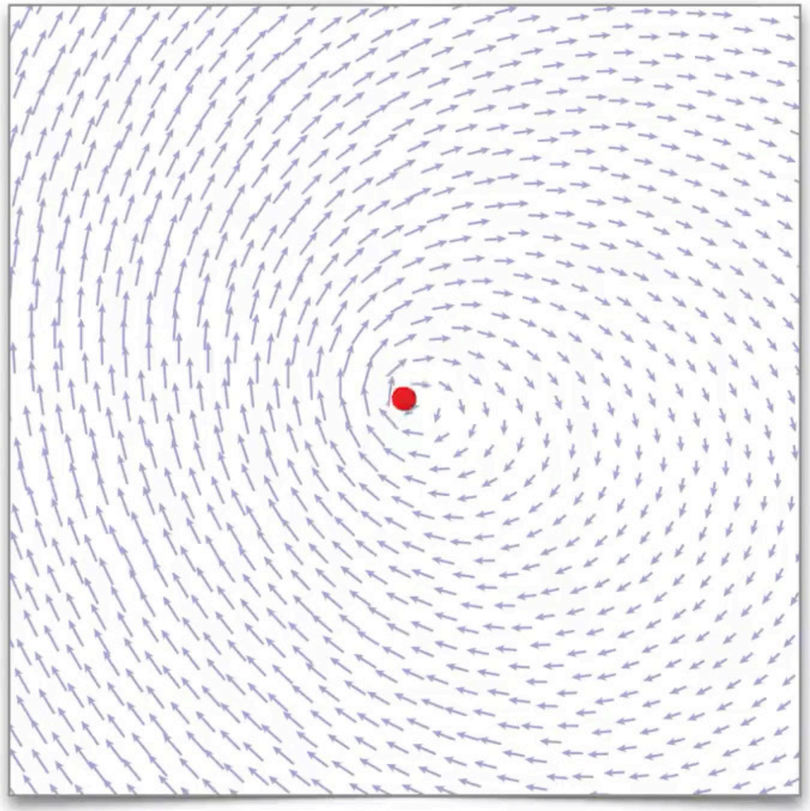
trajectory (X_t)

absorption (σ)

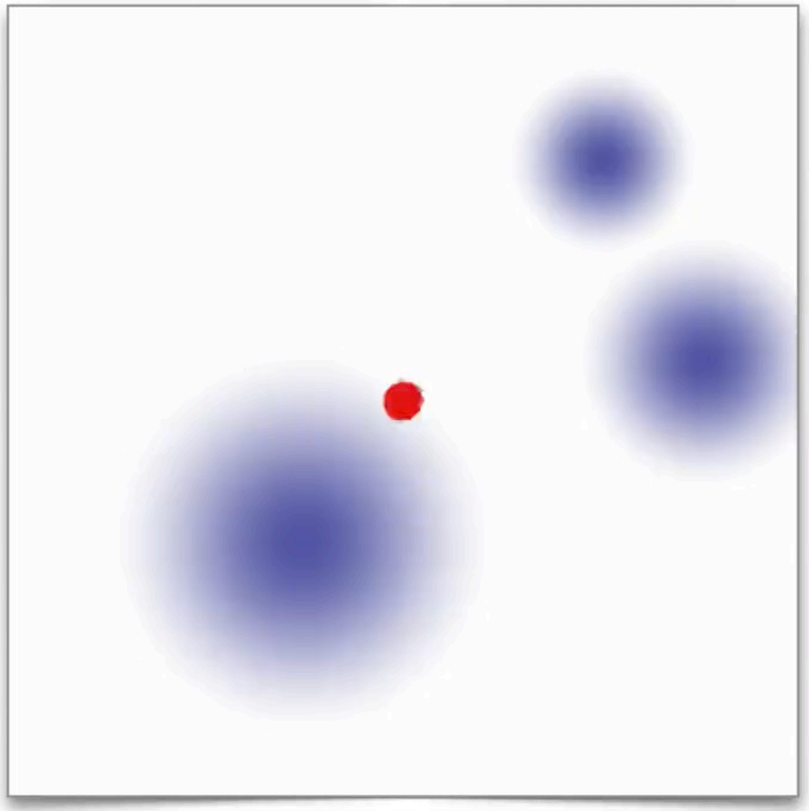
Stochastic differential equations (SDEs)



Brownian motion
with variable diffusion $\alpha(x)$



Brownian motion
with drift $\overrightarrow{\omega}(x)$



Brownian motion
with absorption $\sigma(x)$

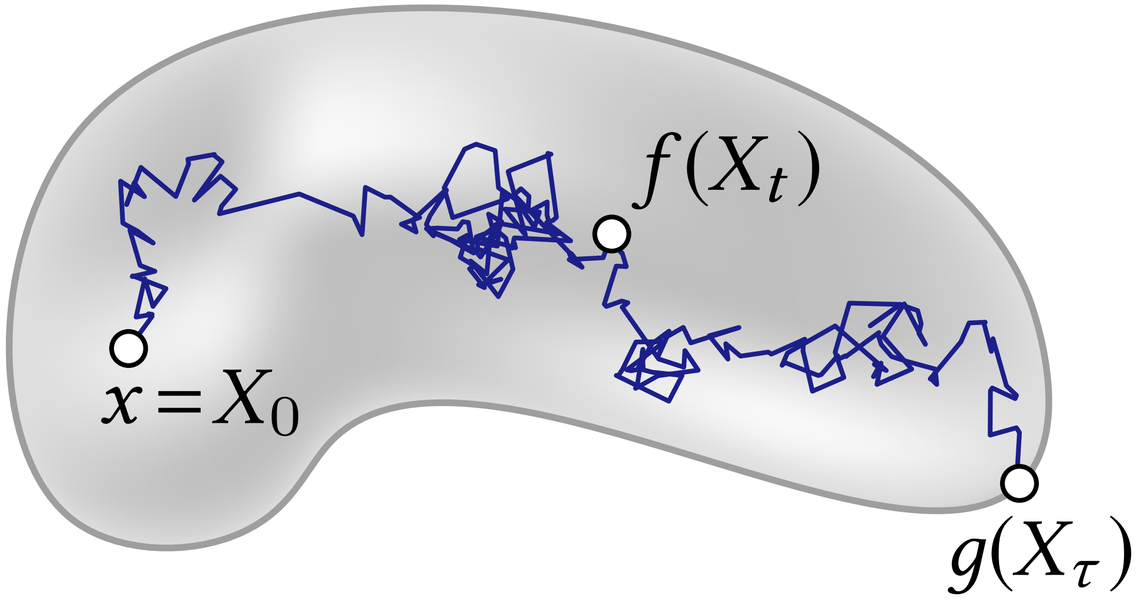
Stochastic representation for variable coefficient PDEs

Feynman Kac formula

$$u(x) = \mathbb{E}\left[\int_0^\tau \overset{\text{absorption}}{e^{-\int_0^t \sigma(X_s)\,ds}}\, f(X_t)\, dt \;+\; \overset{\text{absorption}}{e^{-\int_0^\tau \sigma(X_t)\,dt}}\, g(X_\tau)\right]$$

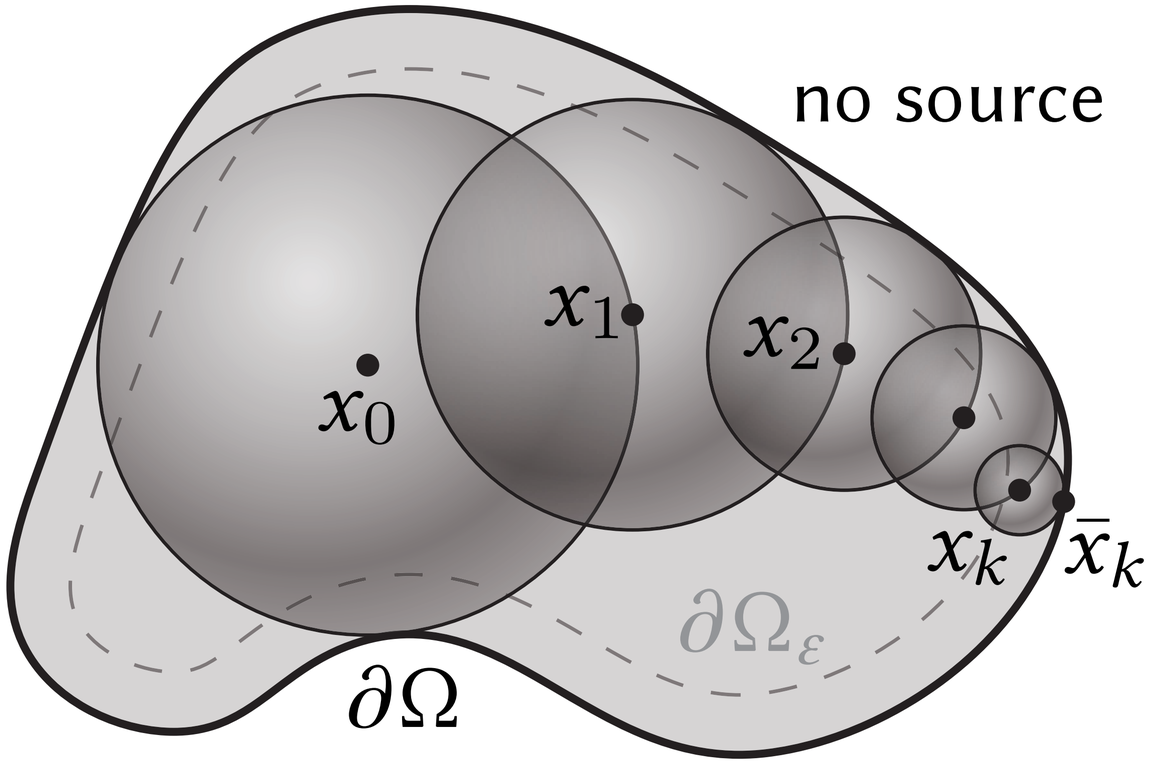
Diffusion process

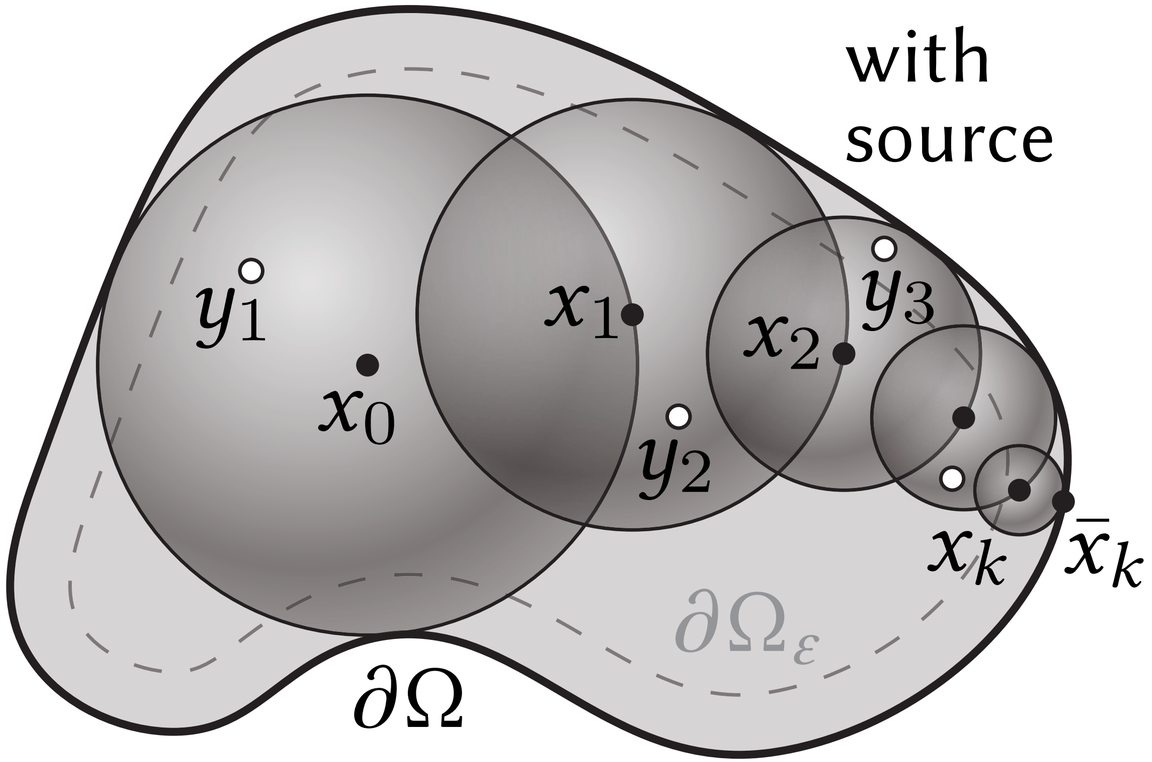
$$dX_t = \underset{\text{drift}}{\vec{\omega}(X_t)}\, dt \;+\; \underset{\text{diffusion}}{\sqrt{\alpha(X_t)}}\, dW_t$$



Walk on Spheres for PDEs with source terms

E.g., $\Delta u = f(x)$; sample the **spatially-varying source** f inside each ball





Transformations to PDE [Sawhney*, Seyb*, Jarosz†, Crane†]

Variable coefficient

$$\nabla \cdot (\alpha \nabla u) + \overrightarrow{\omega} \cdot \nabla u - \sigma\, u = f$$

Girsanov & delta tracking transformations

Constant coefficient

(No approximation!)

$$\Delta u - \bar{\sigma}\, u = \boxed{f(x, \alpha, \overrightarrow{\omega}, \sigma, u)}$$

← recursive

constant

Integral

$$\int_{B(x)} f(y, \alpha, \overrightarrow{\omega}, \sigma, u)\; G^{\bar{\sigma}}(x, y)\; dy + \int_{\partial B(x)} u(z)\; P^{\bar{\sigma}}(x, z)\; dz$$

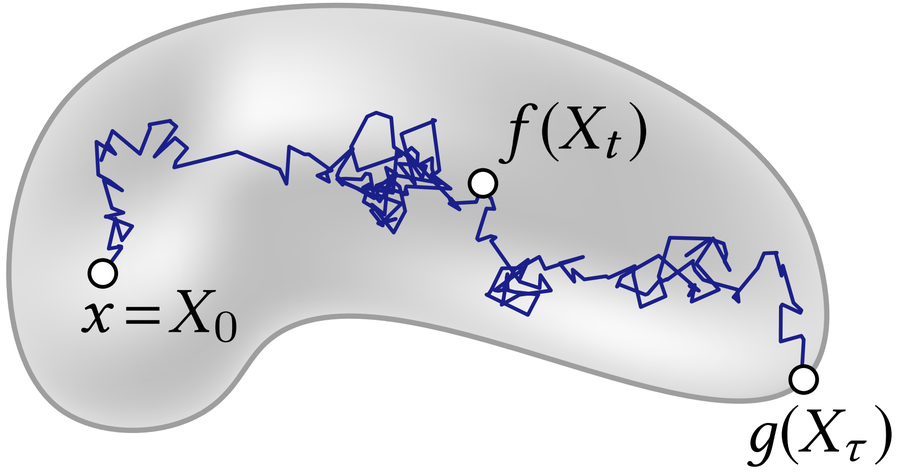
Transformations to Feynman–Kac

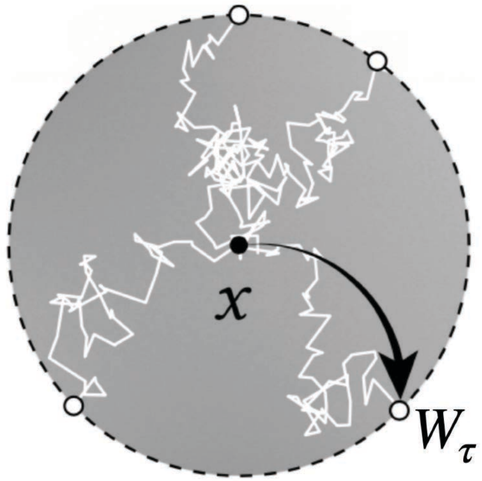
Re-express Feynman Kac in terms of Brownian motion

$$u(x) = \mathbb{E}\left[\int_0^\tau e^{-\int_0^t \sigma(X_s)\,ds} f(X_t)\,dt \;+\; e^{-\int_0^\tau \sigma(X_t)\,dt}\, g(X_\tau)\right]$$

$$dX_t = \vec{\omega}(X_t)\,dt \;+\; \sqrt{\alpha(X_t)}\,dW_t$$

$$dX_t = dW_t$$

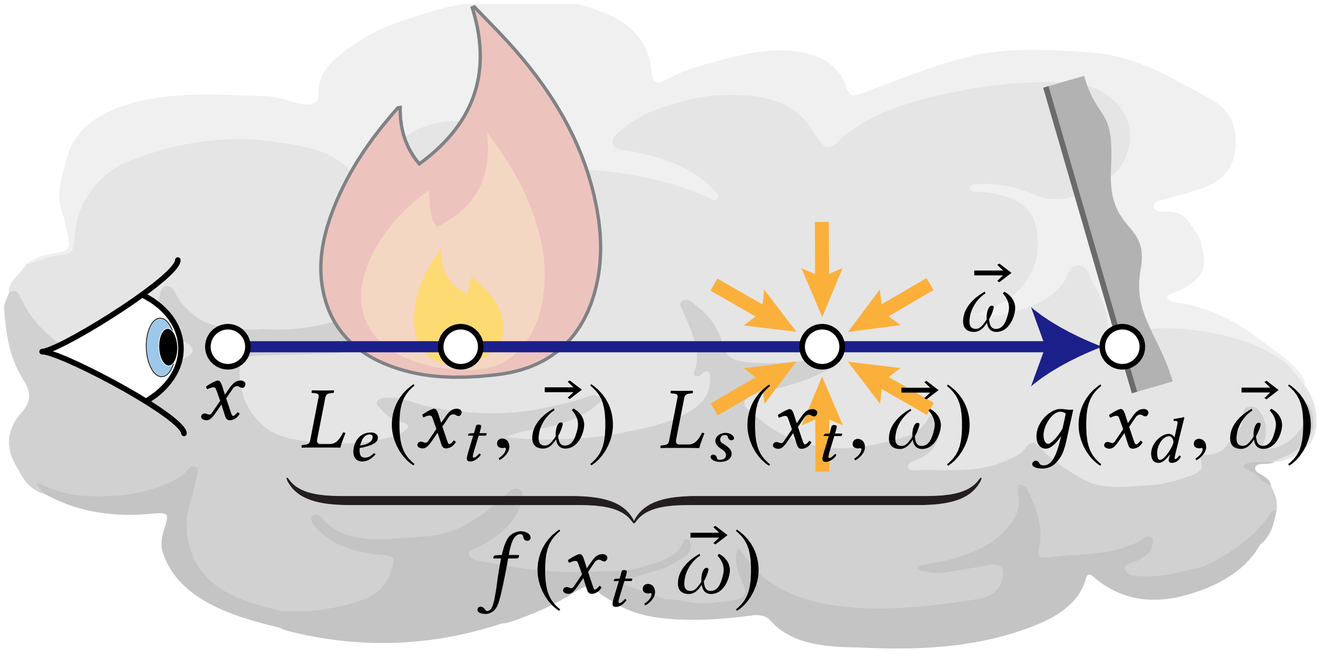




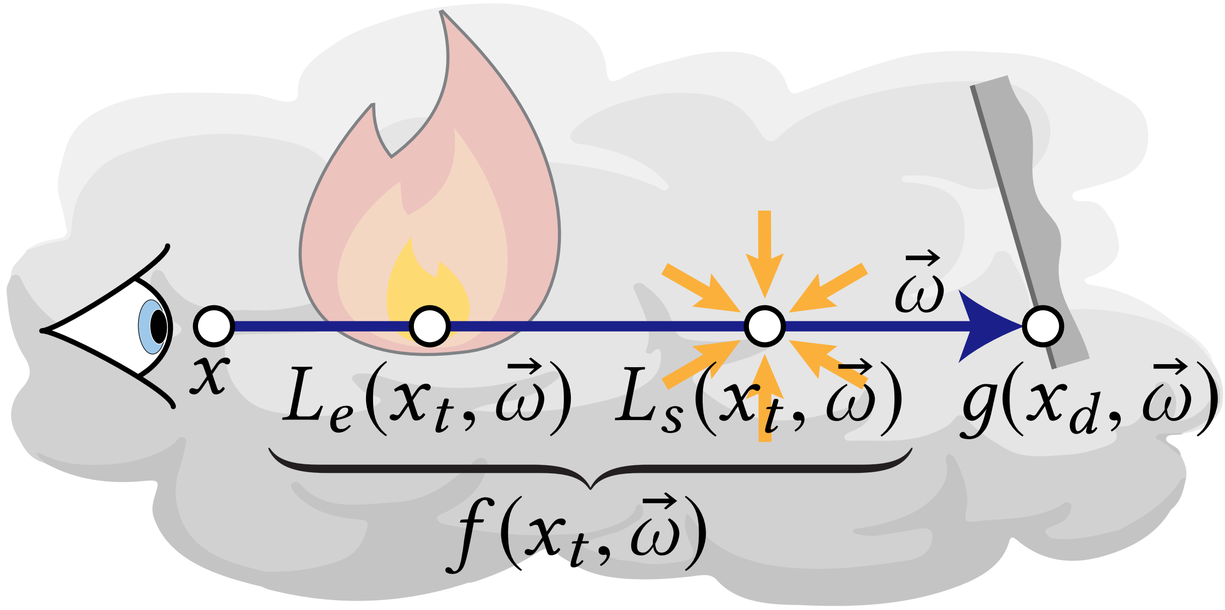
Volume Rendering Equation (VRE)

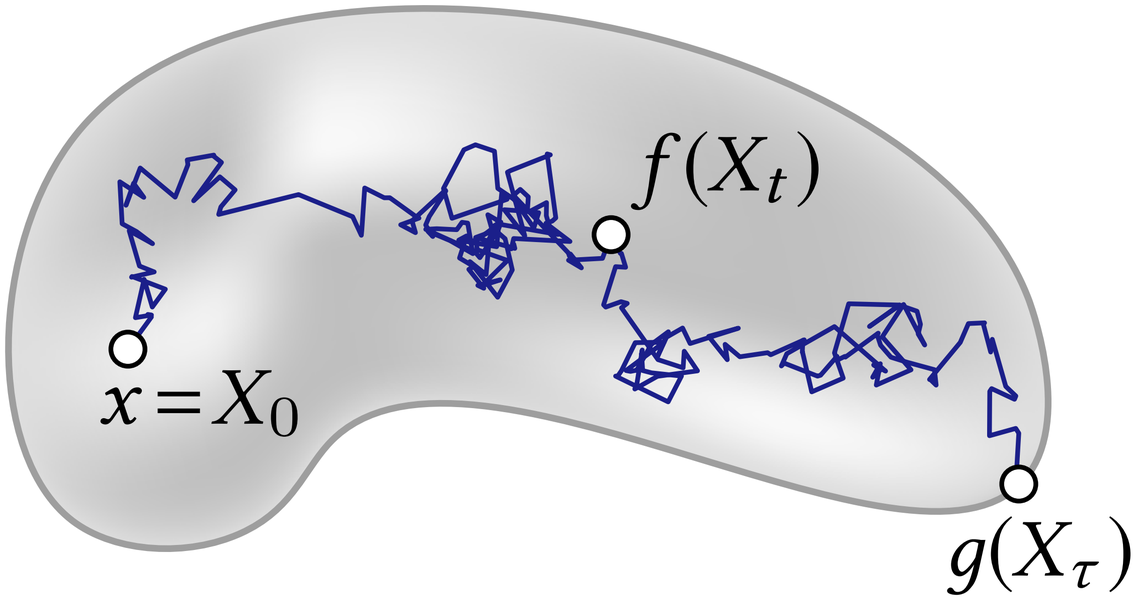
VRE describes the radiance in heterogeneous absorbing & scattering media

$$L(w, \vec{\omega}) = \int_0^d e^{-\int_0^t \sigma(x_s)\, ds} f(x_t, \vec{\omega})\, dt + e^{-\int_0^d \sigma(x_t)\, dt} g(x_d, \vec{\omega})$$



Structural connection between VRE & Feynman–Kac





$$L(w, \vec{\omega}) = \int_0^d e^{-\int_0^t \sigma(x_s)\, ds} f(x_t, \vec{\omega})\, dt + e^{-\int_0^d \sigma(x_t)\, dt} g(x_d, \vec{\omega})$$

VRE gives radiance in heterogeneous absorbing & scattering media

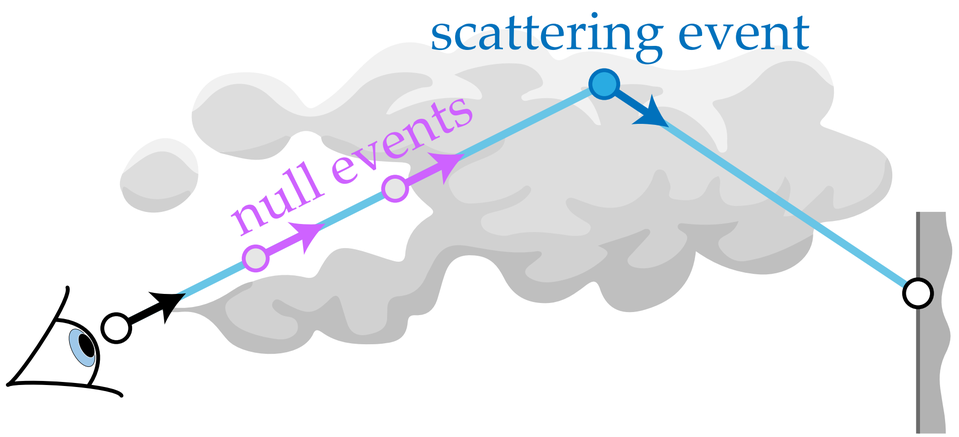
$$u(x) = \mathbb{E}\left[\int_0^\tau e^{-\int_0^t \sigma(W_s)\, ds} f(W_t)\, dt + e^{-\int_0^\tau \sigma(W_t)\, dt} g(W_\tau)\right]$$

Feynman–Kac for 2nd order variable coefficient PDEs

Take inspiration from Volume Rendering

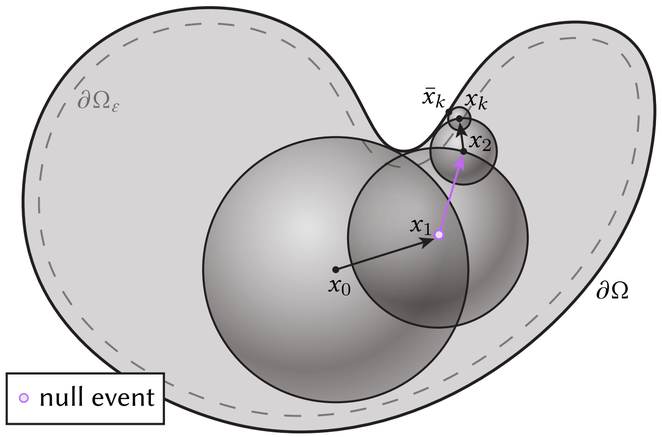
INPUT (4SPP)

DENOISED (4SPP)



DELTA TRACKING (RENDERING)

[Woodcock et al 1965; Raab et al 2008]



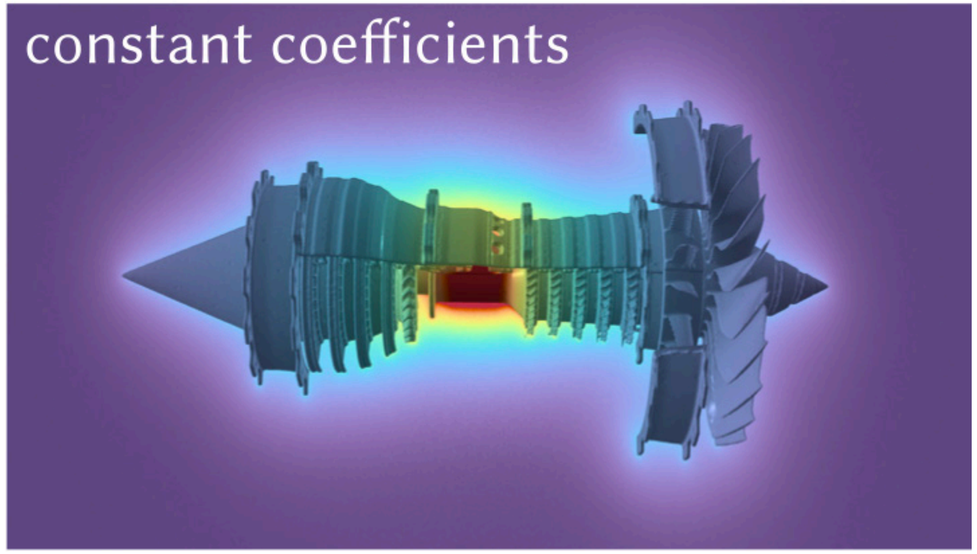
DELTA TRACKING (WOS)

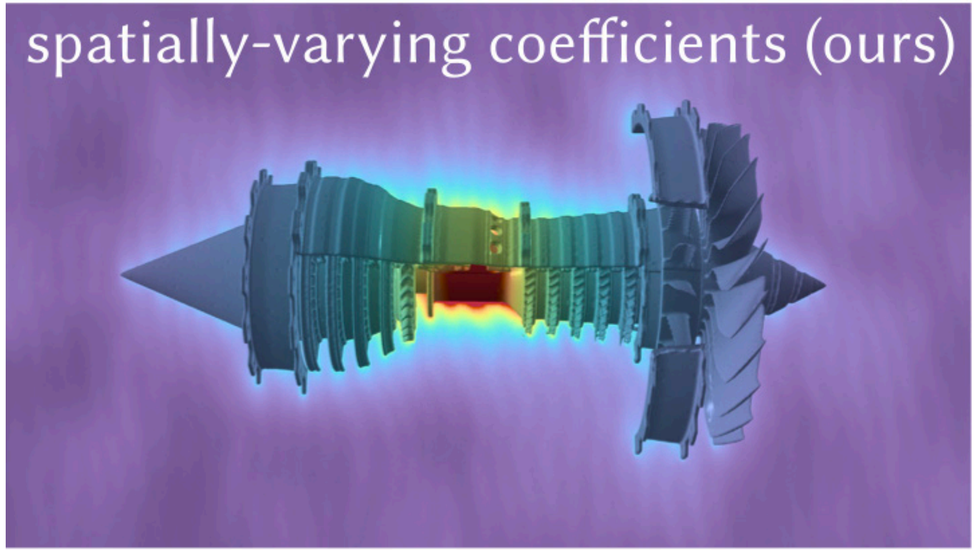
[Sawhney*, Seyb*, Jarosz†, Crane†]

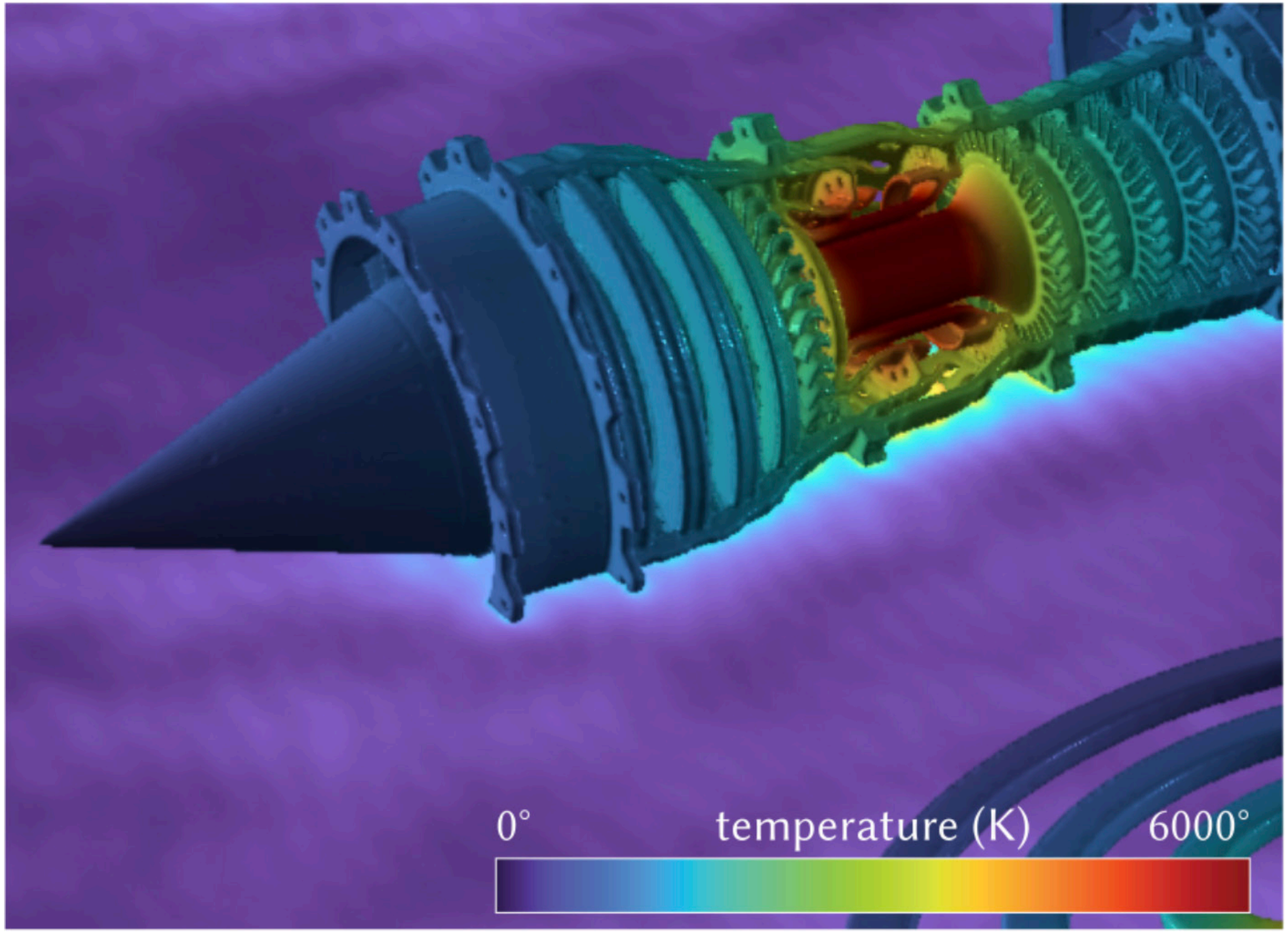
TARGET (4096SPP)

[Hofmann et al, 2021]

To solve PDEs with variable material coefficients



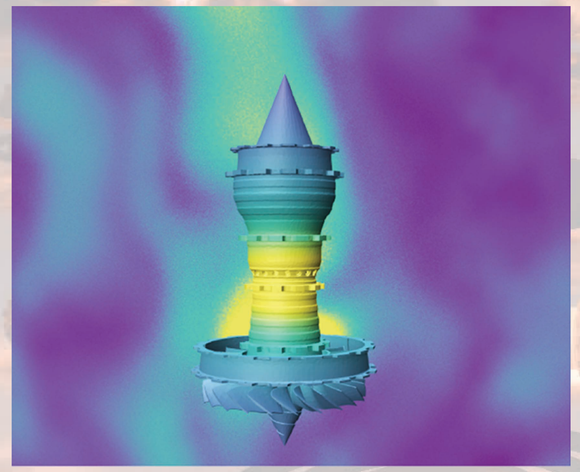




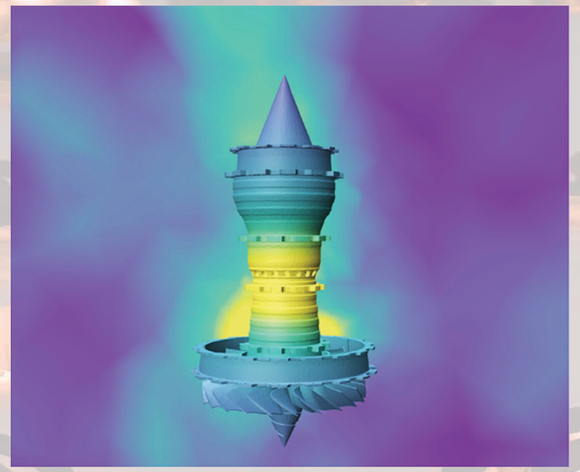

No model cleanup, reduction or homogenization!

Adaptive Mesh Refinement (AMR)

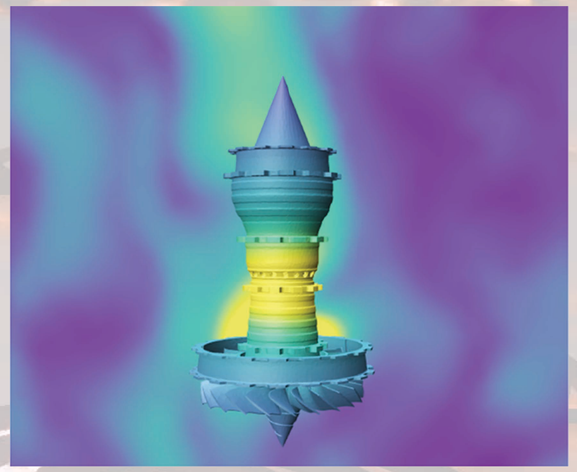
WoS – **10 minutes**

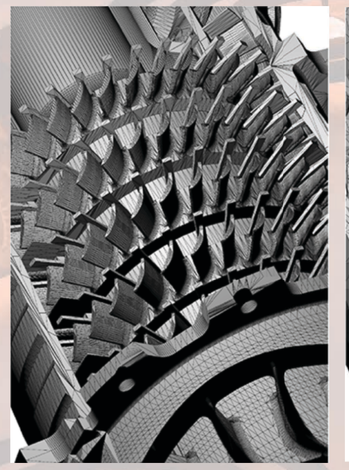


FEM – **1.5 hours**

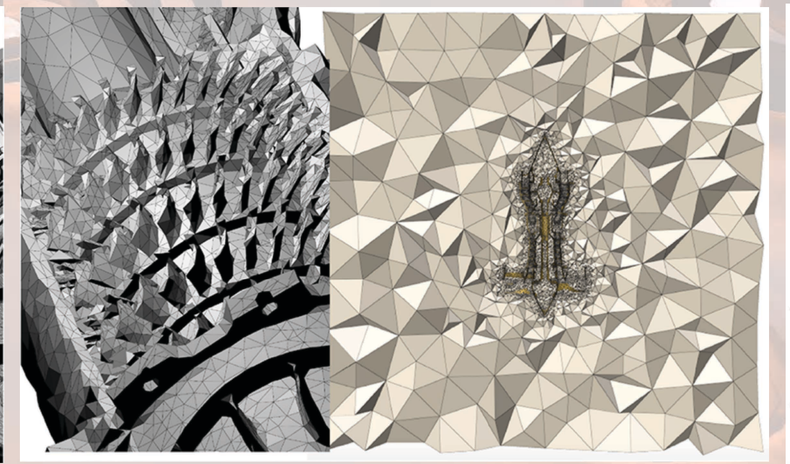


FEM+AMR – **2.5 hours**

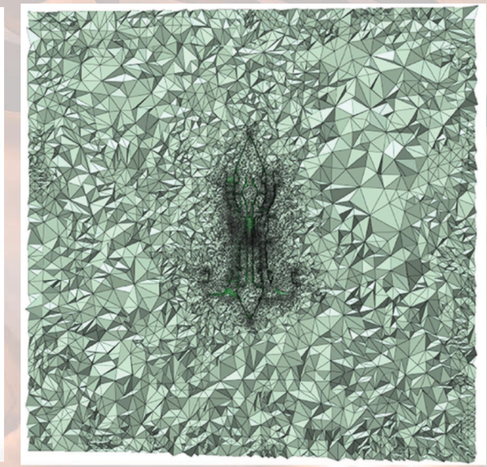




input mesh
(used directly by WoS)



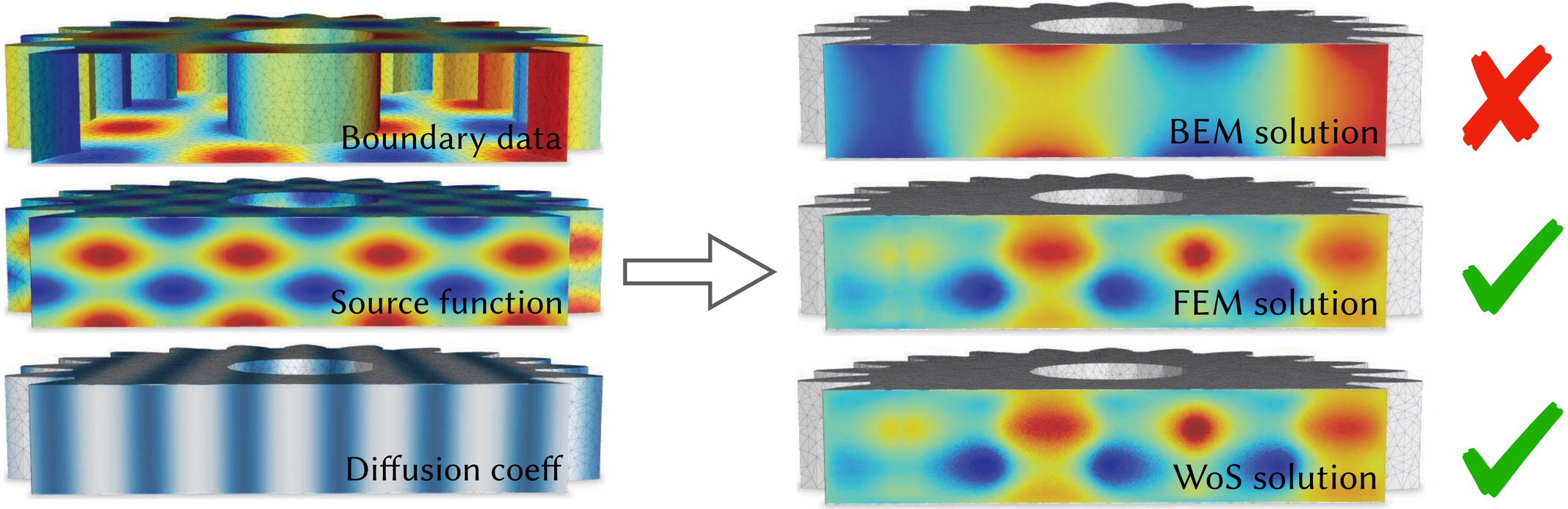
FEM mesh
(~700k tetrahedra)



adaptive FEM mesh
(8.5 million tetrahedra)

Comparison with conventional solvers

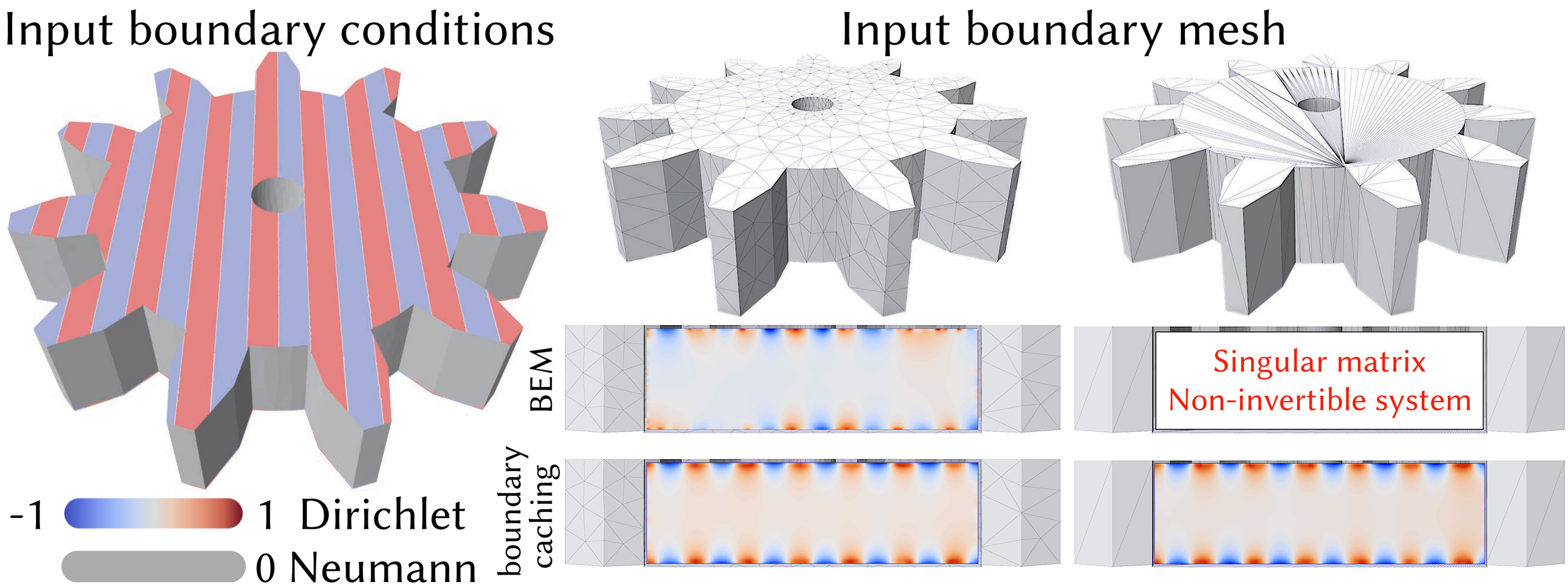
Boundary element method does not require a volume mesh



To handle source terms or variable coefficients, must integrate with FEM

Comparison with conventional solvers

Tessellation independent as boundary samples are generated randomly



Today, walk on spheres can solve…

Laplace equation

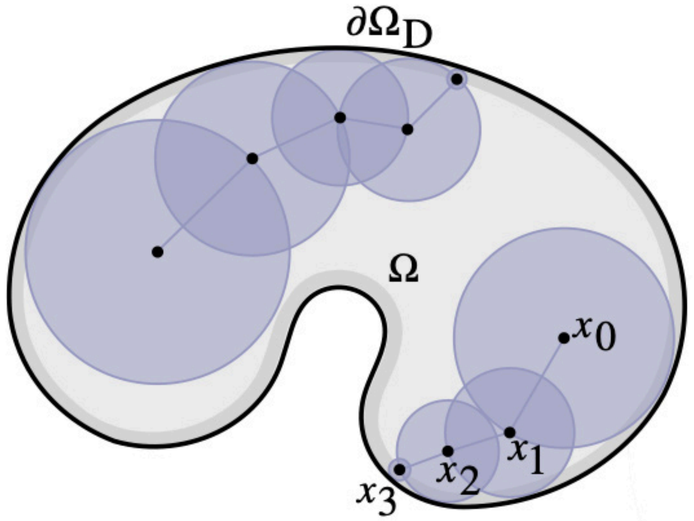
$$\Delta u = 0$$

Poisson equation

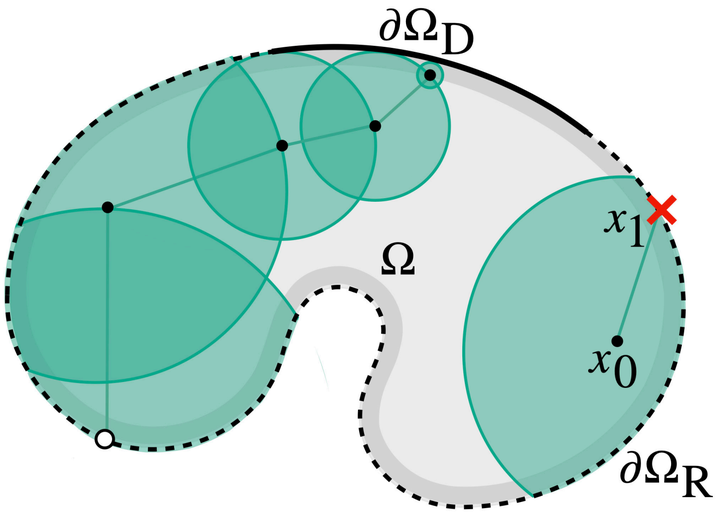
$$\Delta u = f$$

Elliptic equations

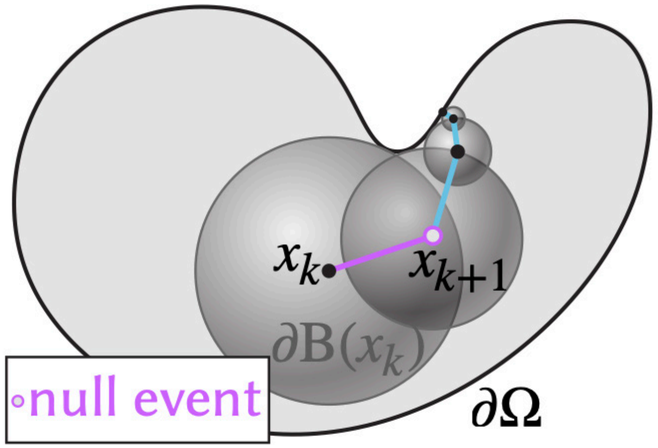
$$\nabla \cdot (\alpha \nabla u) + \vec{\omega} \cdot \nabla u - \sigma u = f$$



walk on spheres



walk on stars



walk on spheres (delta-tracking)

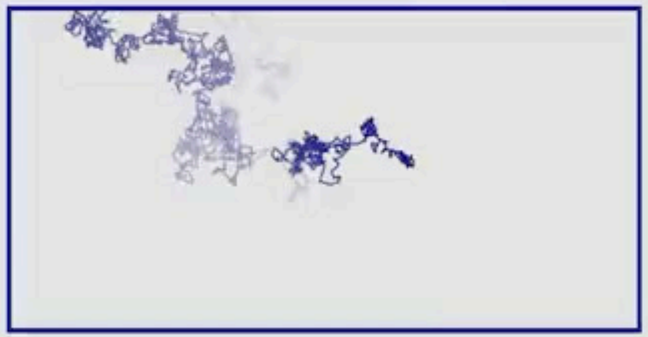
Stochastic Processes and Partial Differential Equations (PDEs)

Brownian motion
$\Longrightarrow$ initial value problems
(e.g., heat equation)

branching random walks
$\Longrightarrow$ semilinear PDEs
(e.g., reaction-diffusion equations)

radially symmetric Levy process
$\Longrightarrow$ fractional order operators
(e.g., fractional Laplace equation)

stopped Brownian motion
$\Longrightarrow$ boundary value problems
(e.g., Laplace equation)



reflected Brownian motion
$\Longrightarrow$ Neumann boundary conditions
(i.e., prescribe normal derivative)

partially reflected Brownian motion
$\Longrightarrow$ Robin boundary conditions
(i.e., prescribe derivative + value)

[animation credit: Keenan Crane]

Boundary Integral Equation

For a Poisson equation $\Delta u = f$, we can express solution as:

$$u(x) = \int_{\partial\Omega} P(x,y)\, u(y)\, dy - \int_{\partial\Omega} G(x,y) \frac{\partial u}{\partial n_y}\, dy + \int_{\Omega} G(x,y)\, f(y)\, dy$$

BIEs also known for:

– Heat eq
– Helmholtz eq
– Wave eq
– Biharmonic eq
– Linear elasticity
– …

PART 6:
Variance Reduction

Efficiency of Monte Carlo

Squared error of Monte Carlo estimator is given by:

$$\frac{V[f]}{N}$$

$V[f]$ ← variance of the integrand

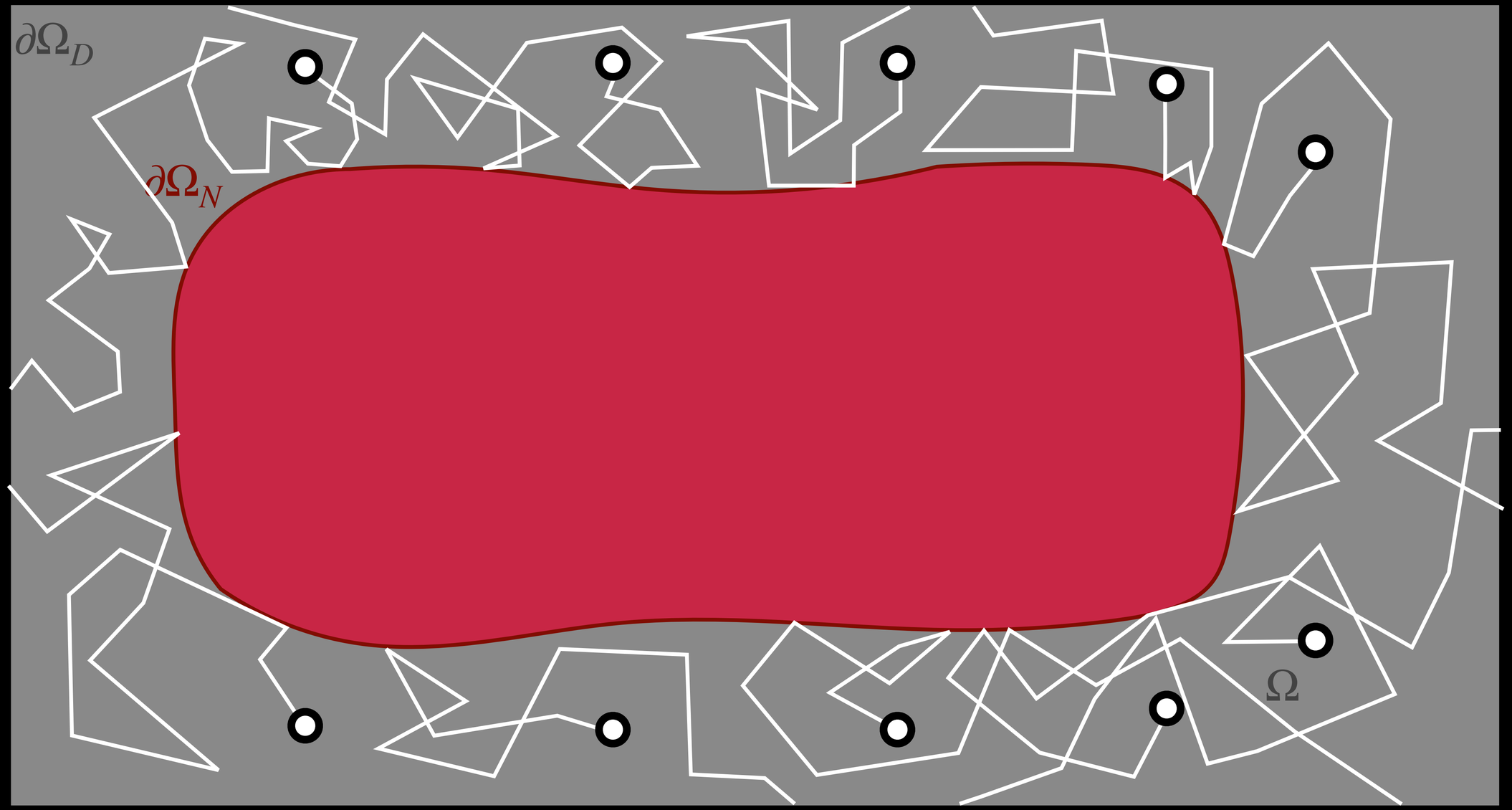
N ← number of samples

To do better, make at least one of these factors smaller:

– **less error for equal time ($V[f]$)**, e.g., importance sampling

– **more samples per second ($1/N$)**, e.g., parallelism, GPUs, caching

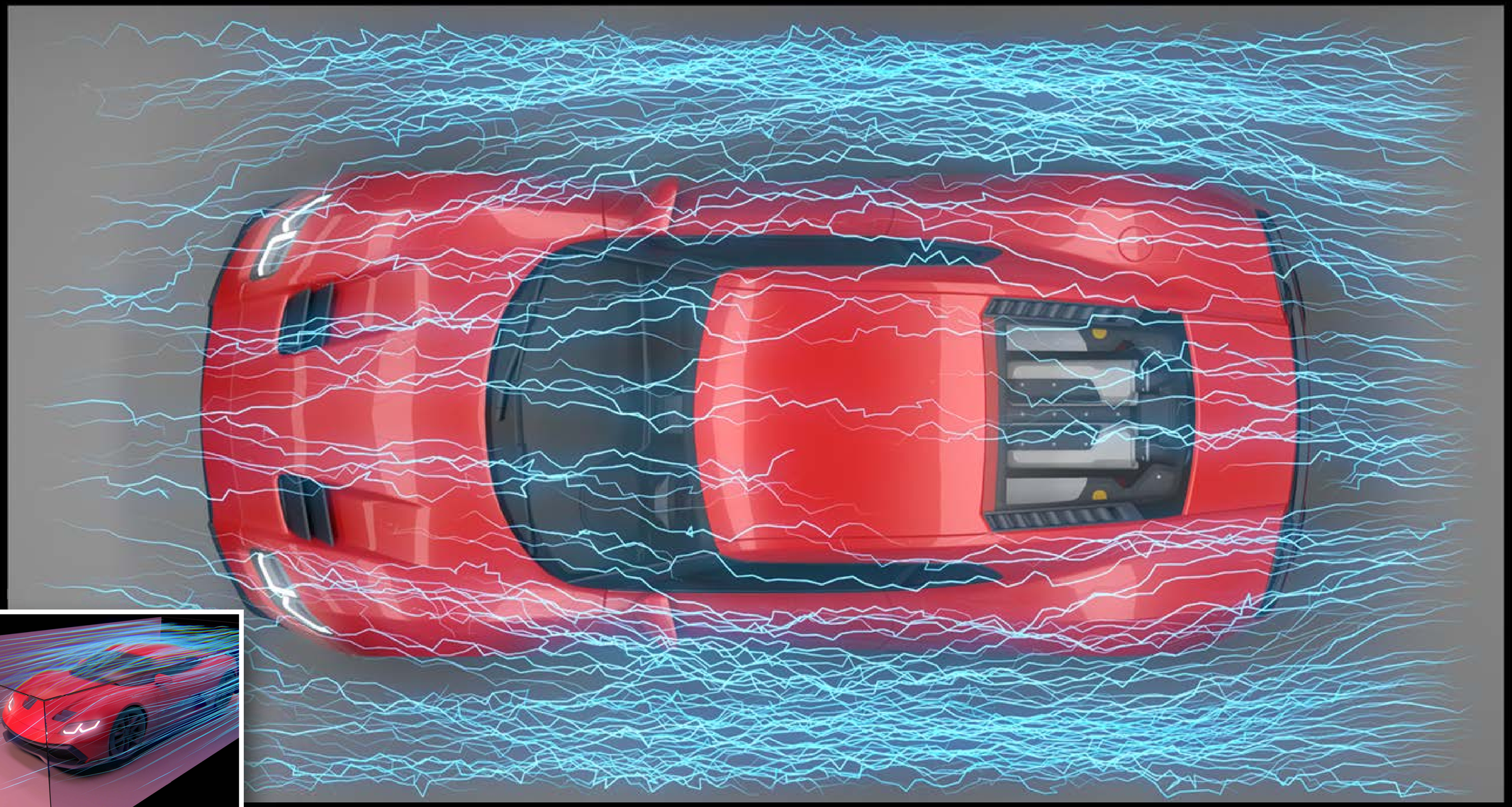
Solution evaluated independently at very point – parallelism

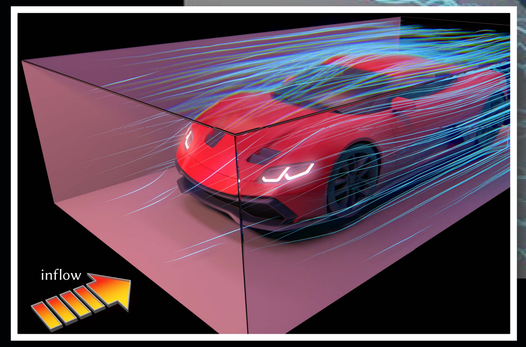


walk on stars

Solution evaluated independently at every point – redundancy

No system of linear equations!

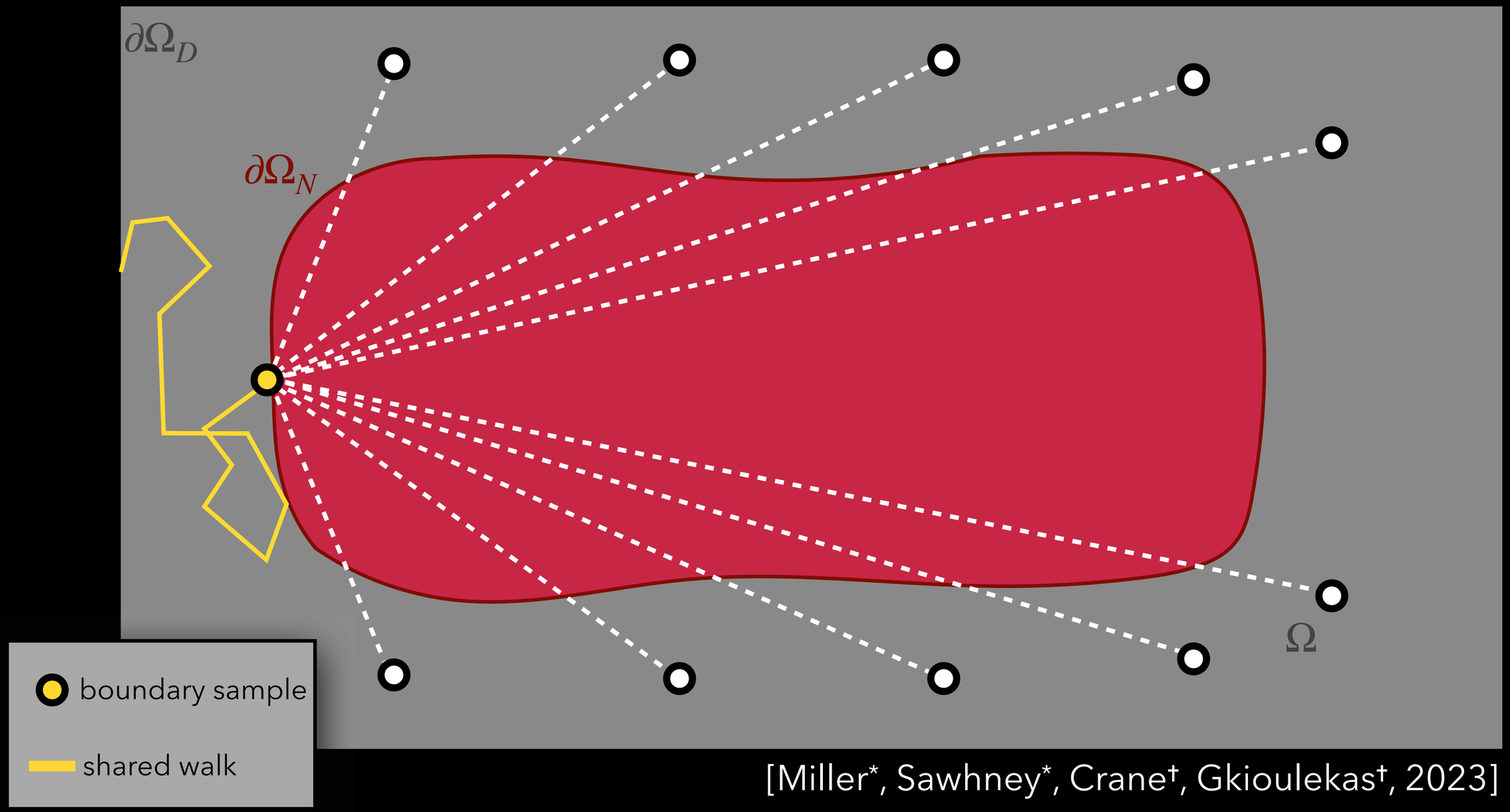




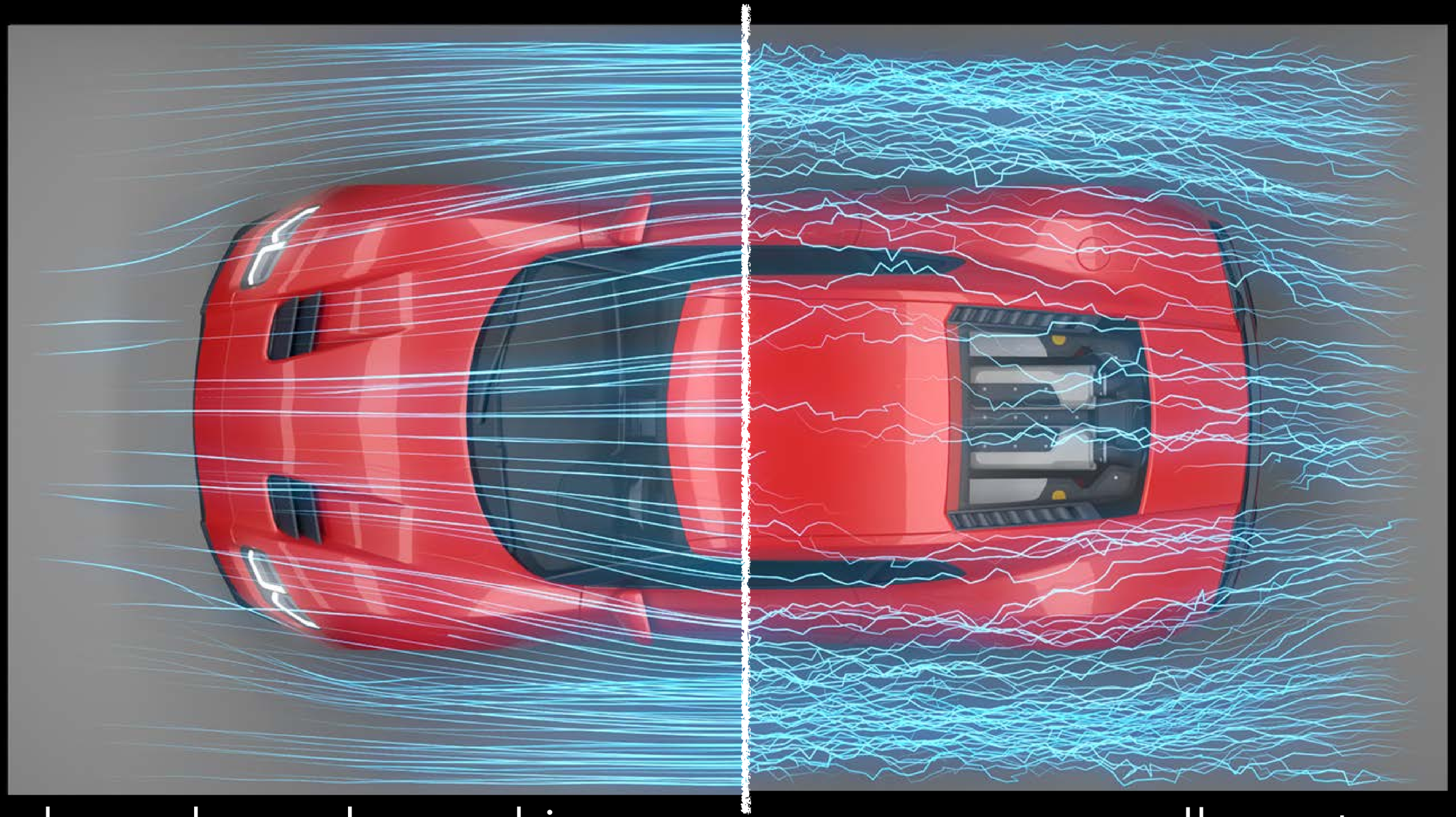
wind tunnel

walk on stars

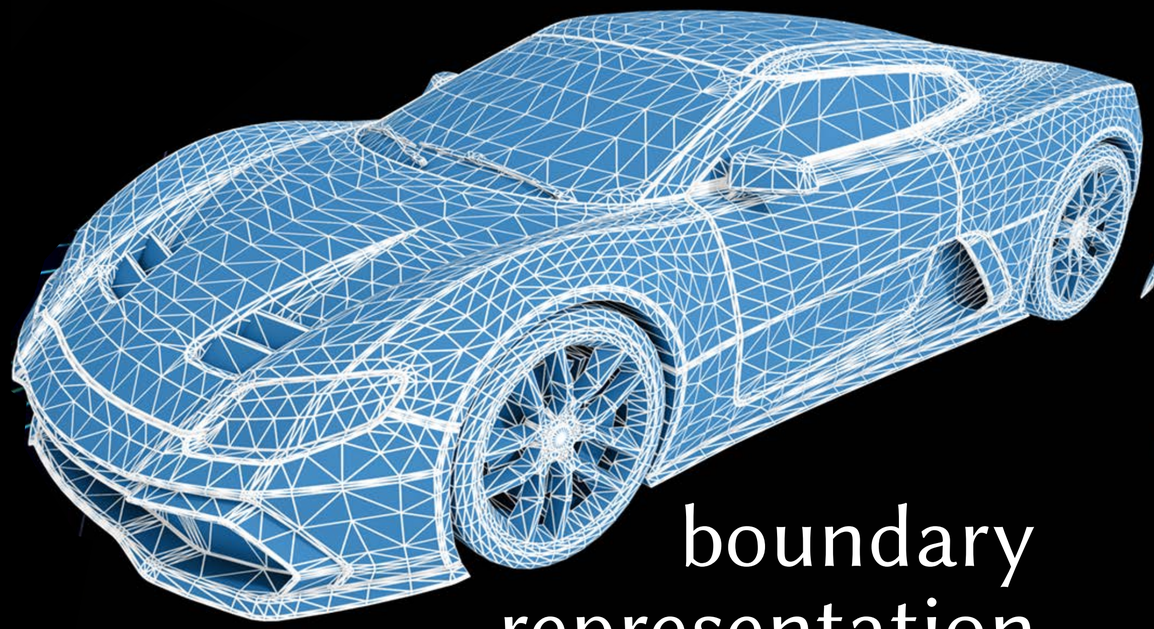
boundary value caching



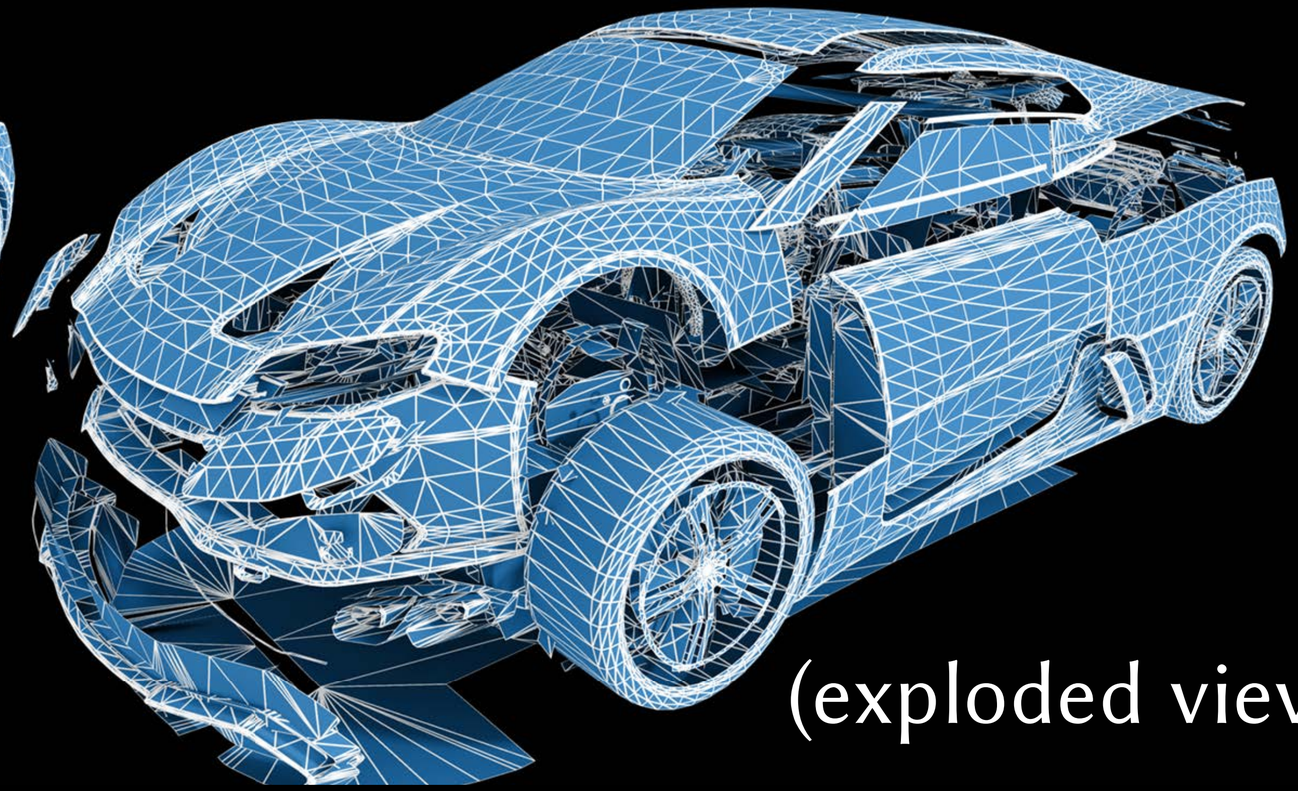
[Miller*, Sawhney*, Crane†, Gkioulekas†, 2023]


boundary value caching
walk on stars

Robustly handle meshes intended for visualization



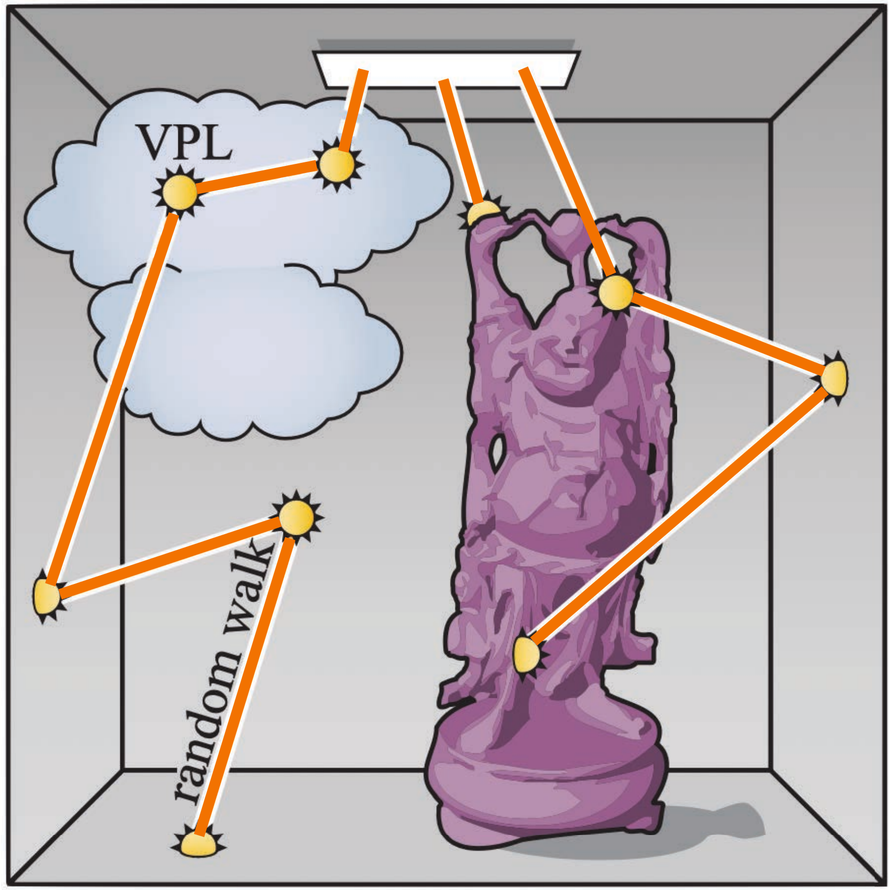
boundary representation



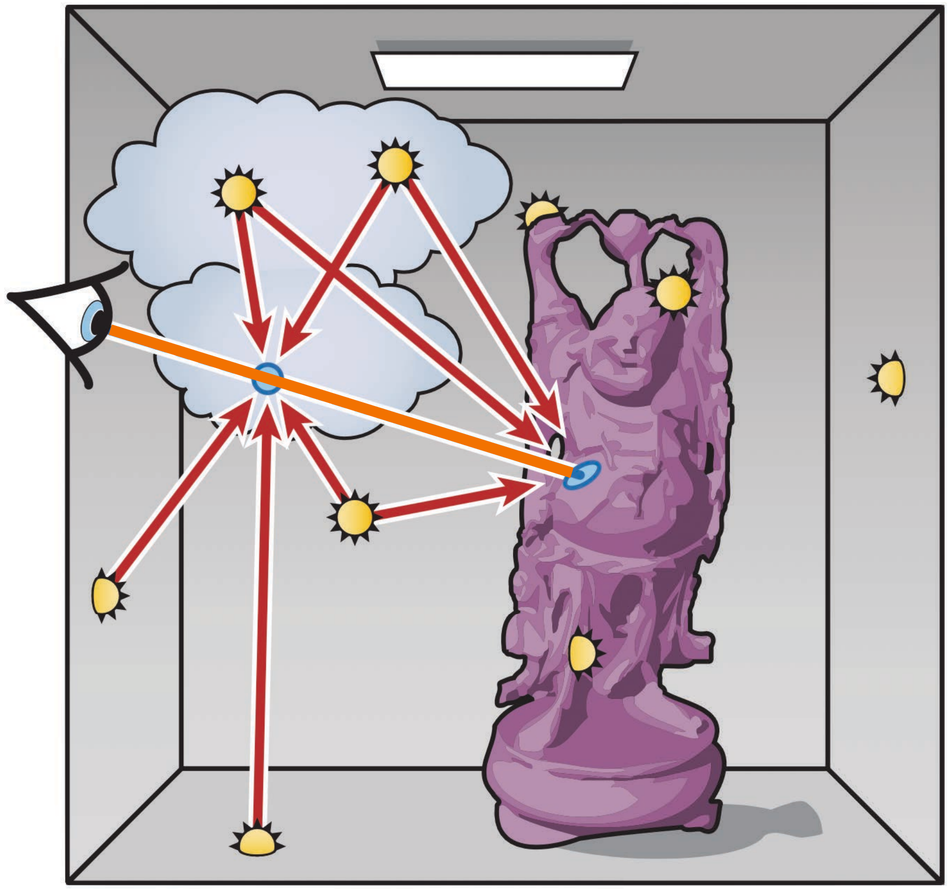
(exploded view)

Sample reuse in Monte Carlo Ray Tracing

Virtual Point Light Methods (VPLs)



Step 1: Deposit radiance estimates



Step 2: Reuse cached radiance estimates

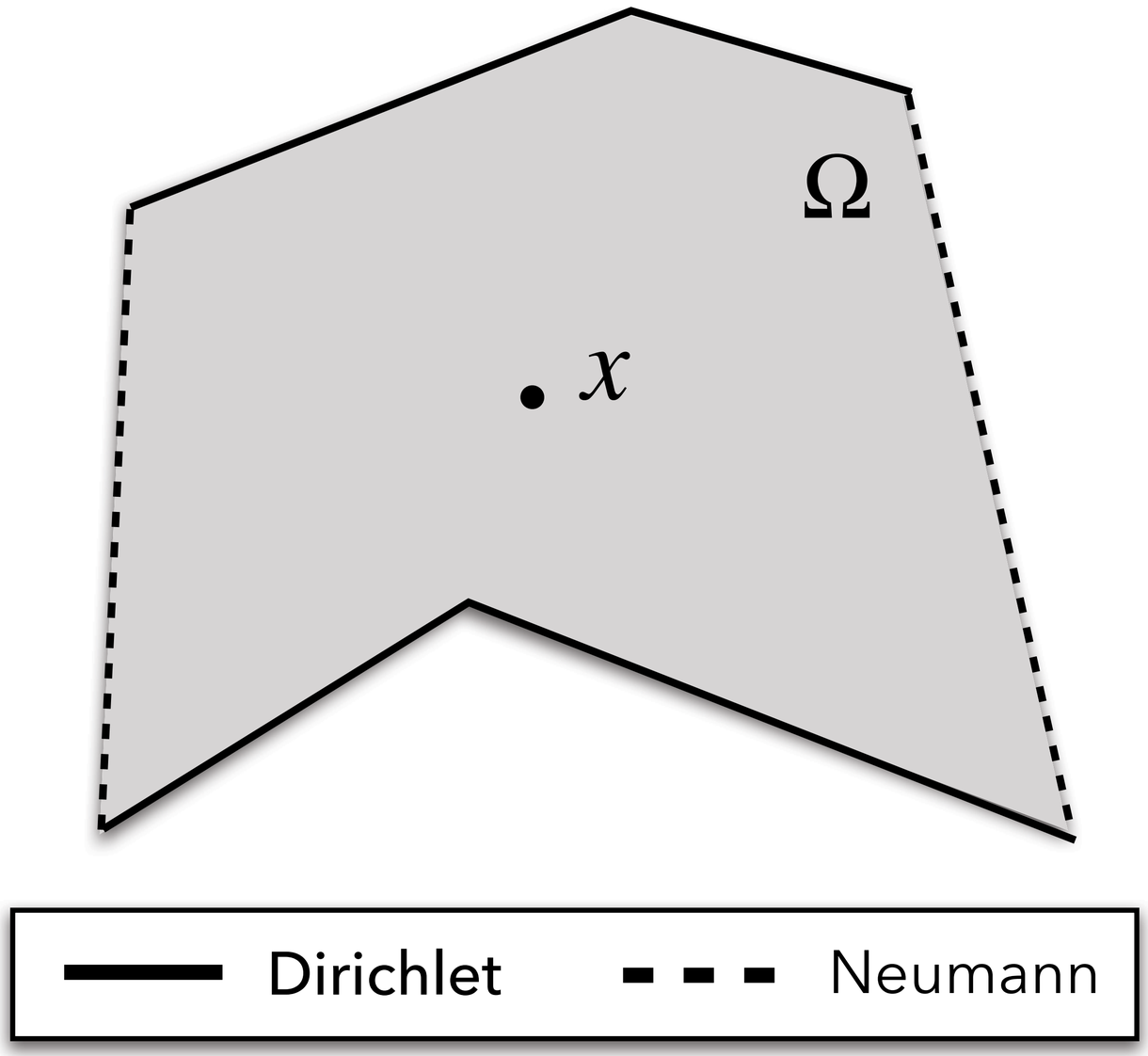
Sample reuse for PDEs

Laplace equation

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

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

$$\frac{\partial u}{\partial n} = h \quad \text{on } \partial\Omega_N$$



Boundary Integral Equation

free-space Poisson kernel

Dirichlet

free-space Green's function

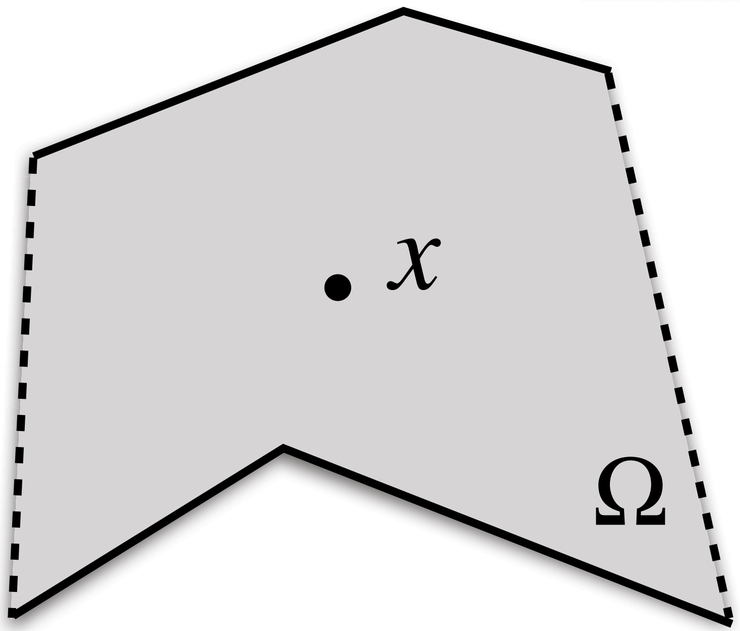
Neumann

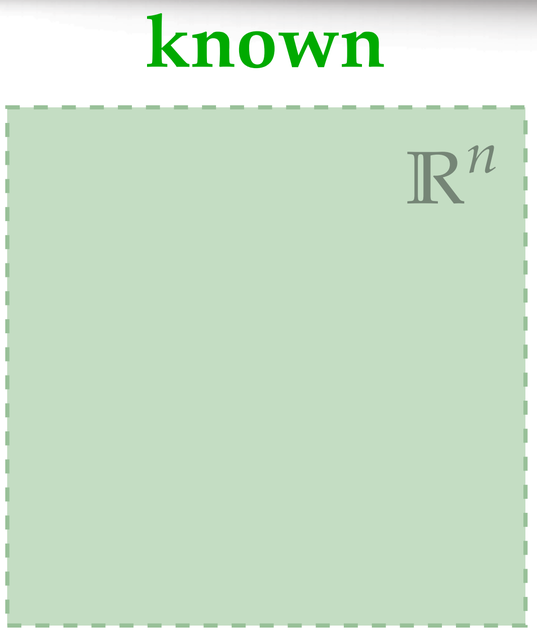
$$u(x) = \int_{\partial\Omega} \frac{\partial G(x,y)}{\partial n} u(y) - G(x,y) \frac{\partial u(y)}{\partial n} \, dy$$

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

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

$$\frac{\partial u}{\partial n} = h \quad \text{on } \partial\Omega_N$$





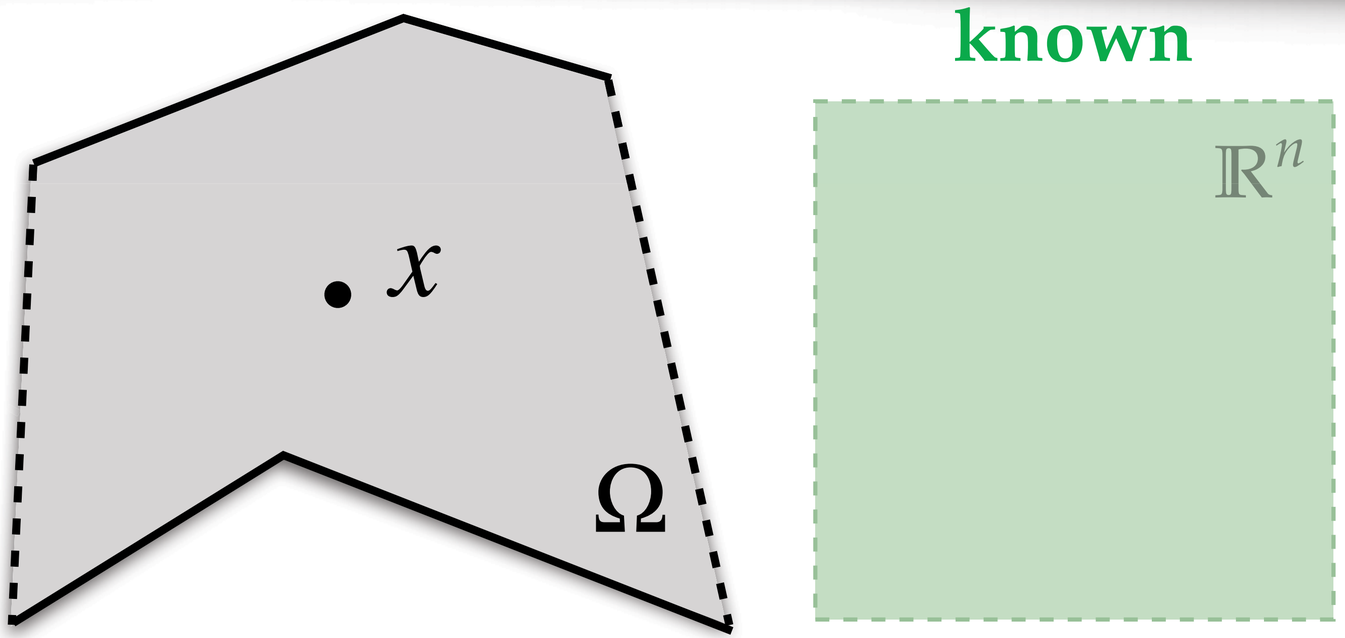
Monte Carlo estimator for BIE

$$\hat{u}(x) = \frac{|\partial\Omega|}{N} \sum_{i=1}^{N} \frac{\partial G(x, y_i)}{\partial n} u(y_i) - G(x, y_i) \frac{\partial u(y_i)}{\partial n}$$

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

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

$$\frac{\partial u}{\partial n} = h \quad \text{on } \partial\Omega_N$$



Monte Carlo estimator for BIE

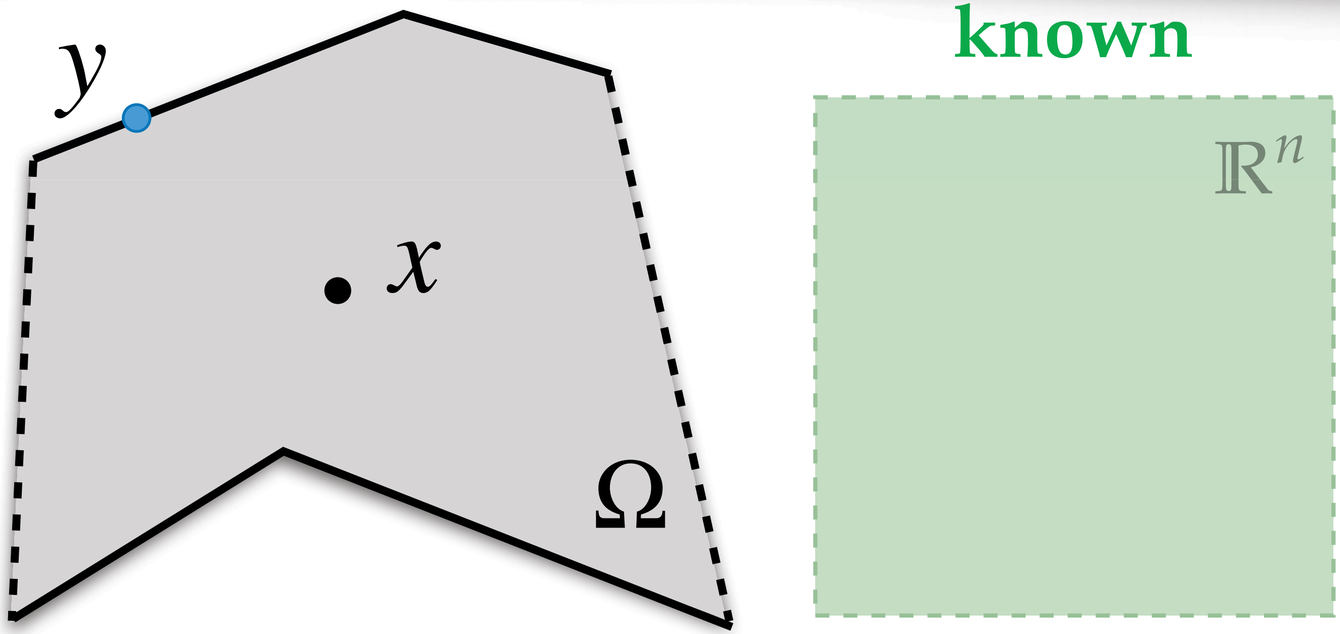
known estimate

$$\widehat{u}(x) = \frac{|\partial\Omega|}{N} \sum_{i=1}^{N} \frac{\partial G(x, y_i)}{\partial n} g(y_i) - G(x, y_i) \frac{\widehat{\partial u(y_i)}}{\partial n}$$

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

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

$$\frac{\partial u}{\partial n} = h \quad \text{on } \partial\Omega_N$$



Monte Carlo estimator for BIE

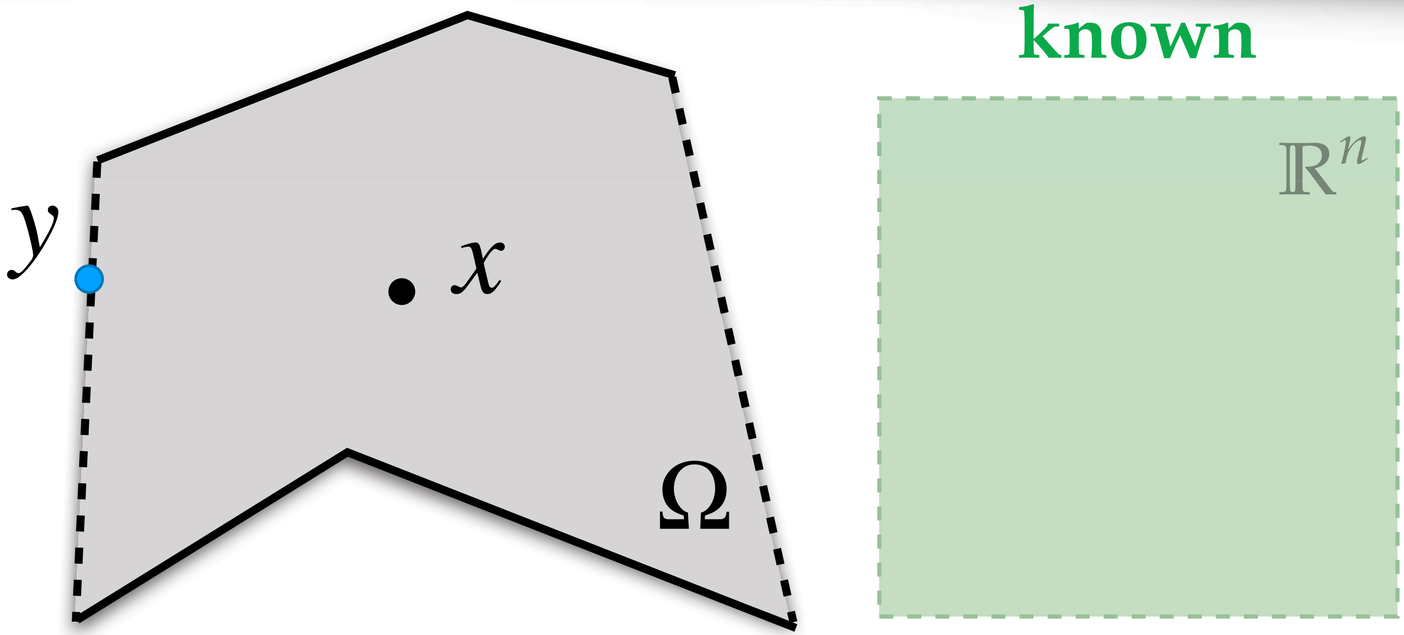
estimate known

$$\hat{u}(x) = \frac{|\partial\Omega|}{N} \sum_{i=1}^{N} \frac{\partial G(x, y_i)}{\partial n} u(y_i) - G(x, y_i)\, h(y_i)$$

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

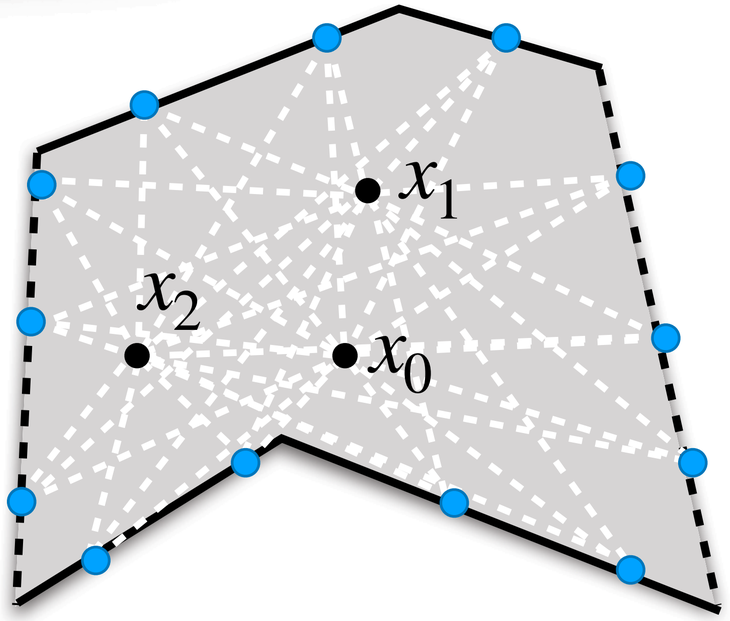
$$u = g \quad \text{on } \partial\Omega_D$$

$$\frac{\partial u}{\partial n} = h \quad \text{on } \partial\Omega_N$$



Monte Carlo estimator for BIE

$$\hat{u}(x) = \frac{|\partial\Omega|}{N} \sum_{i=1}^{N} \frac{\partial G(x, y_i)}{\partial n} \hat{u}(y_i) - G(x, y_i) \frac{\widehat{\partial u(y_i)}}{\partial n}$$



can reuse boundary estimates at ***any*** point in domain

i.e., can estimate many integrals with one set of estimates

Normal derivative estimation

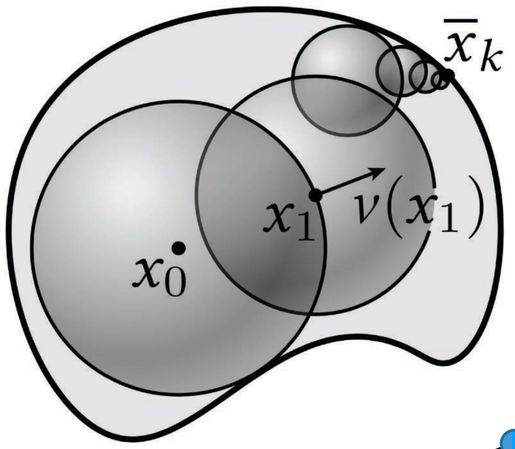
Spatial derivative **inside ball**:

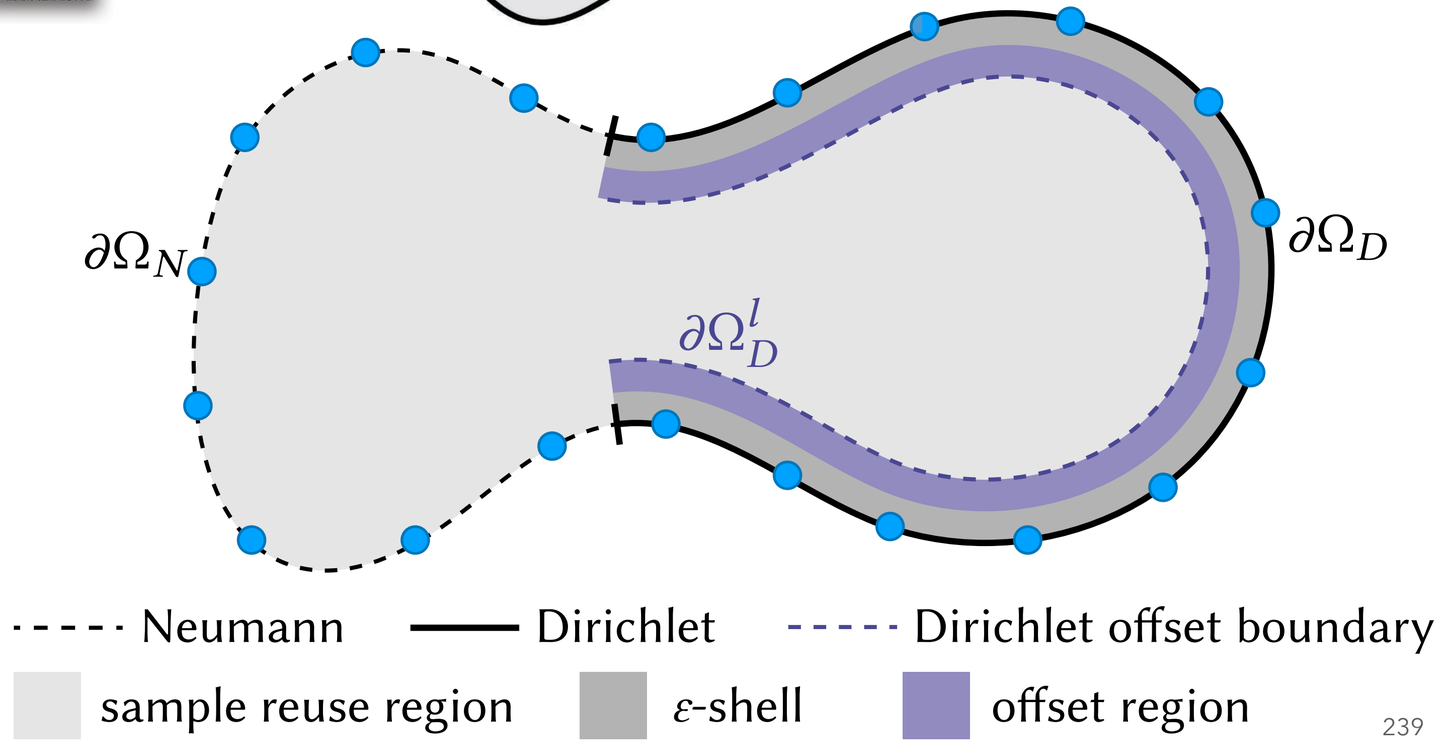
$$\nabla_x u(x) = \frac{1}{|B|} \int_{\partial B} u(y)\, \underbrace{v(y)}_{\text{normal}}\, dy$$

[Sawhney & Crane 2020]

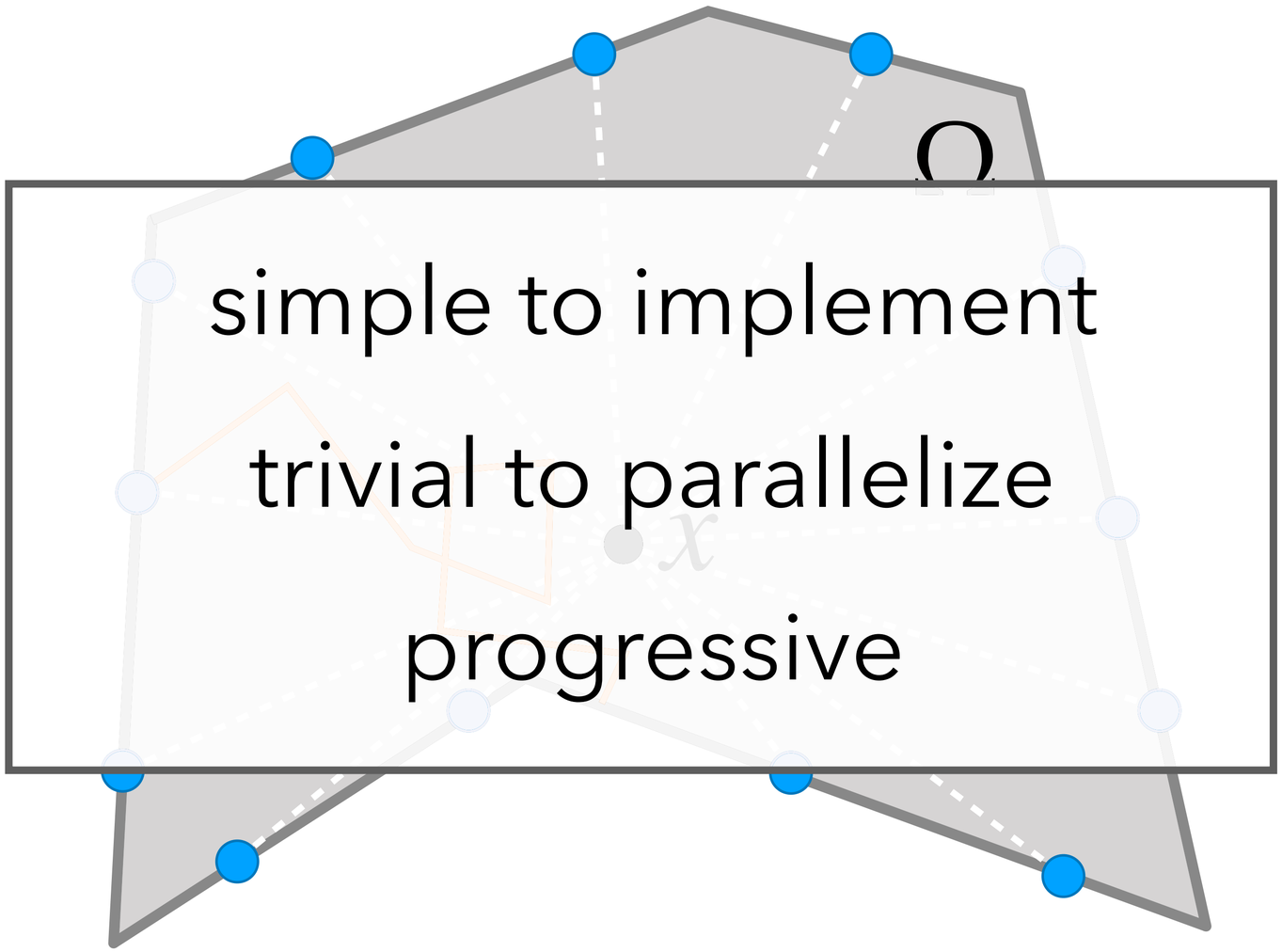
Normal derivative **on boundary**:

$$\frac{du(x)}{dn_x} = n_x \cdot \nabla_x u(x)$$





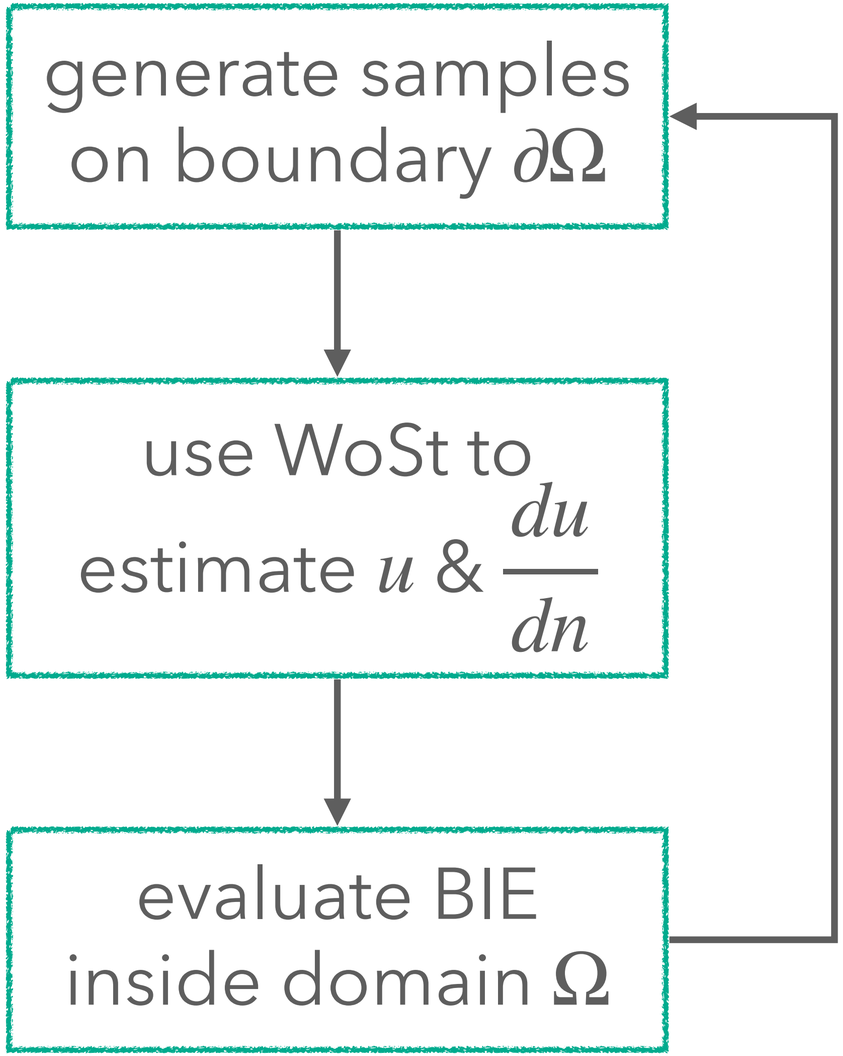
Boundary Value Caching (BVC)



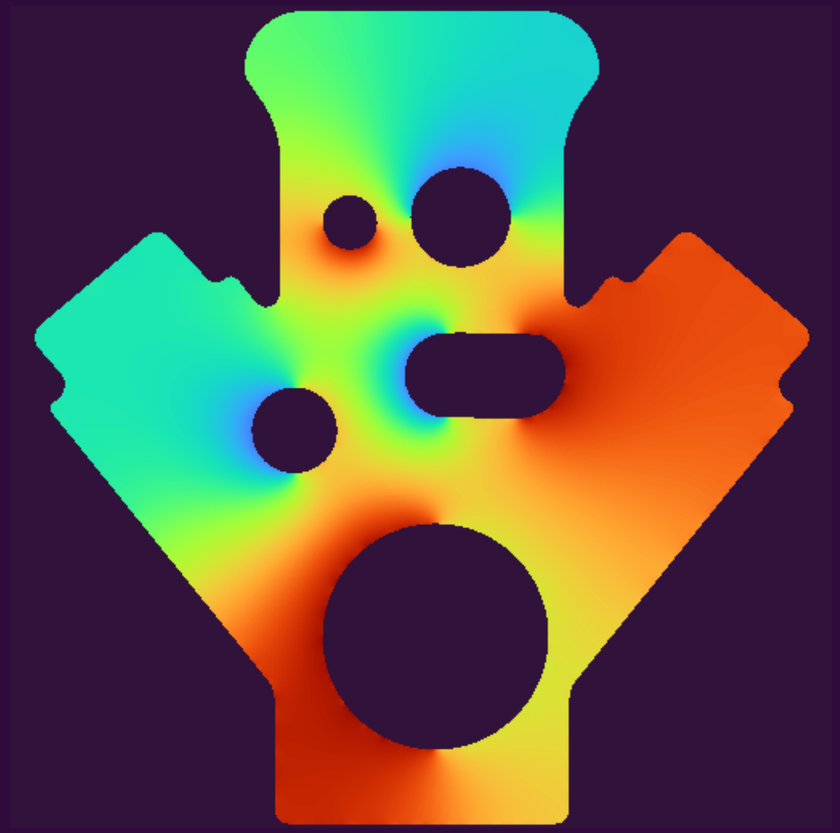
simple to implement

trivial to parallelize

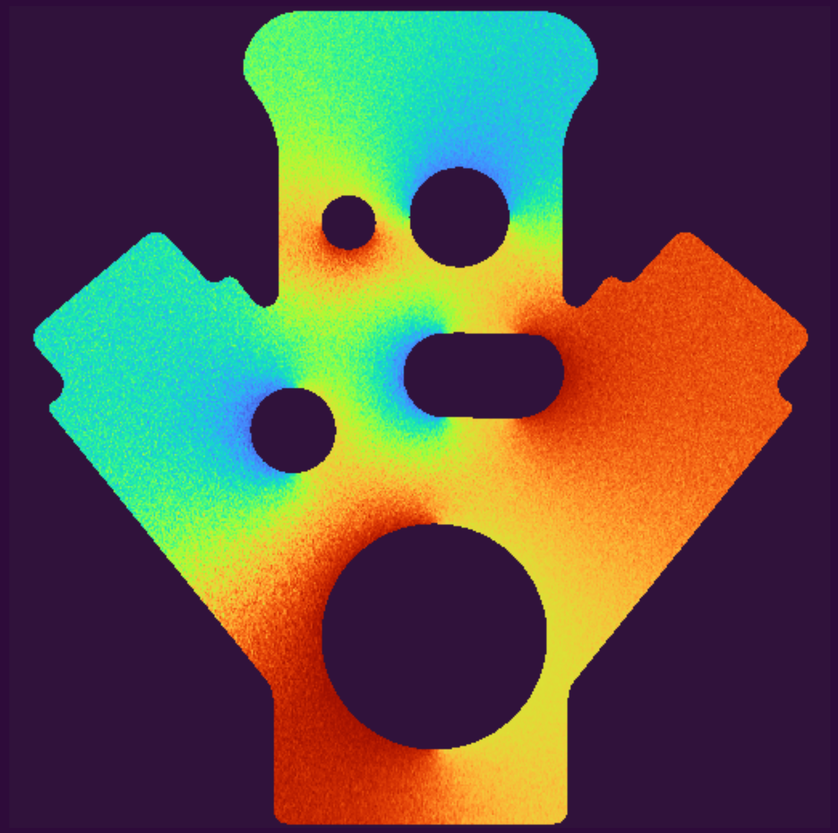
progressive



Benefits of BVC



boundary value caching



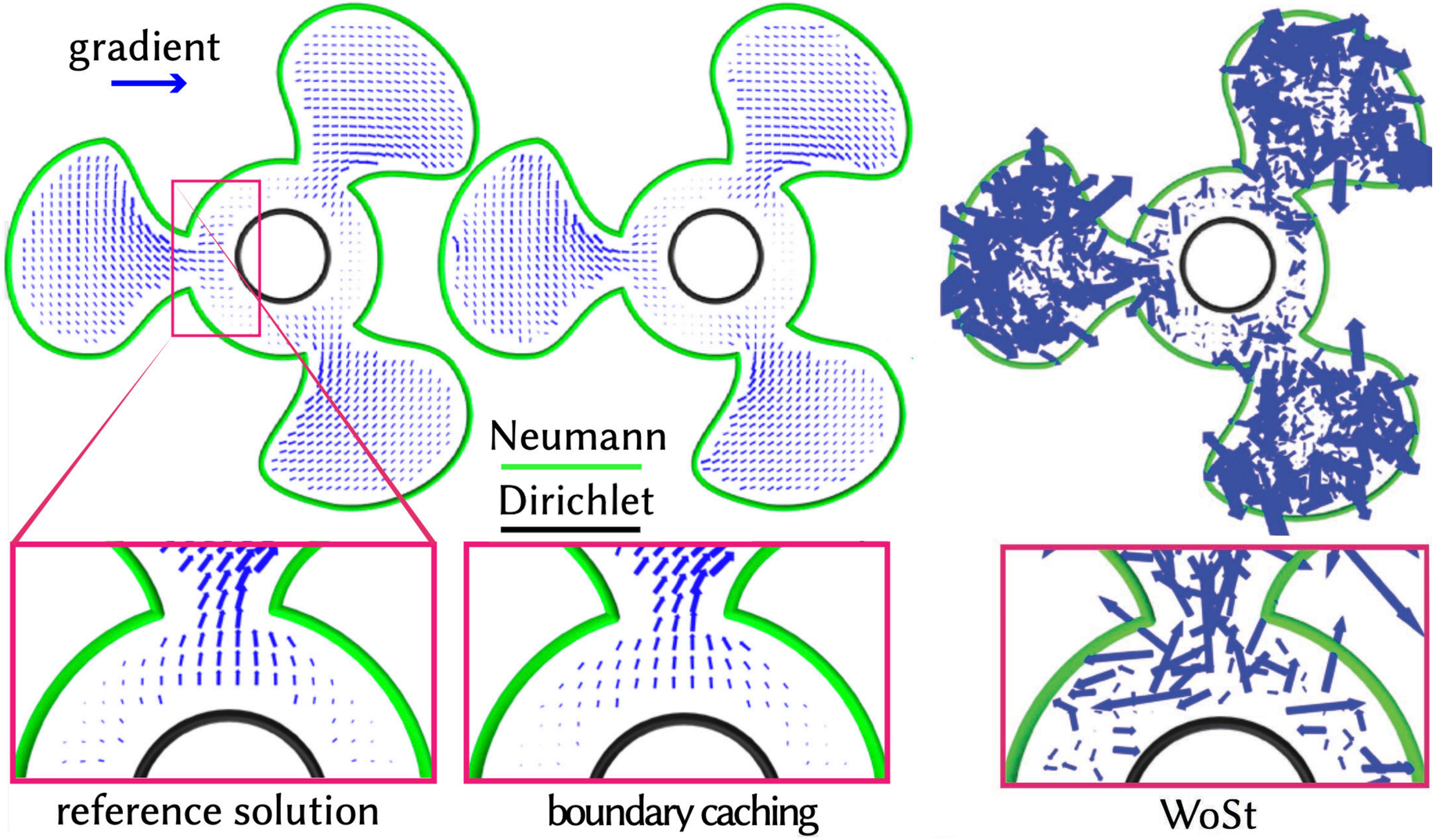
walk on stars

Improved run-time efficiency (sharing global information)

Suppressed noise (due to correlation)

Gradient estimates with BVC

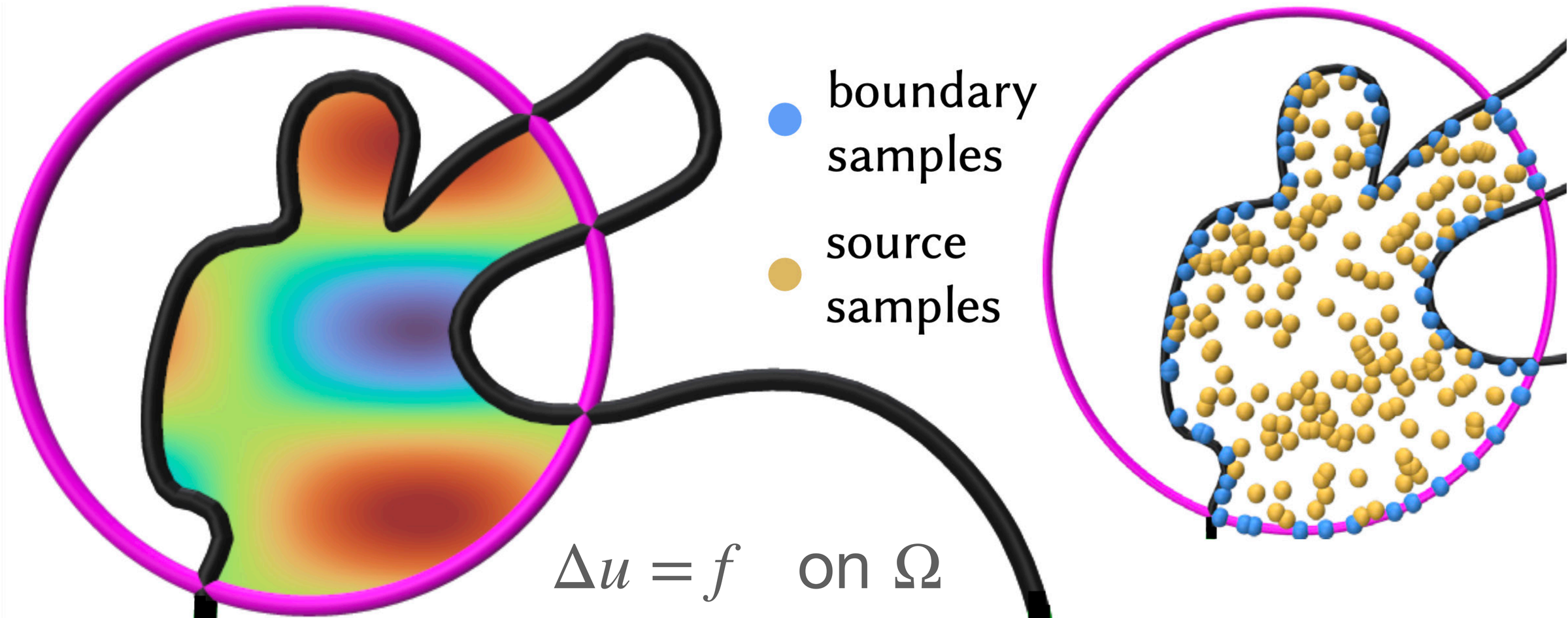
Can reuse **same** boundary cache for gradients!



Source term

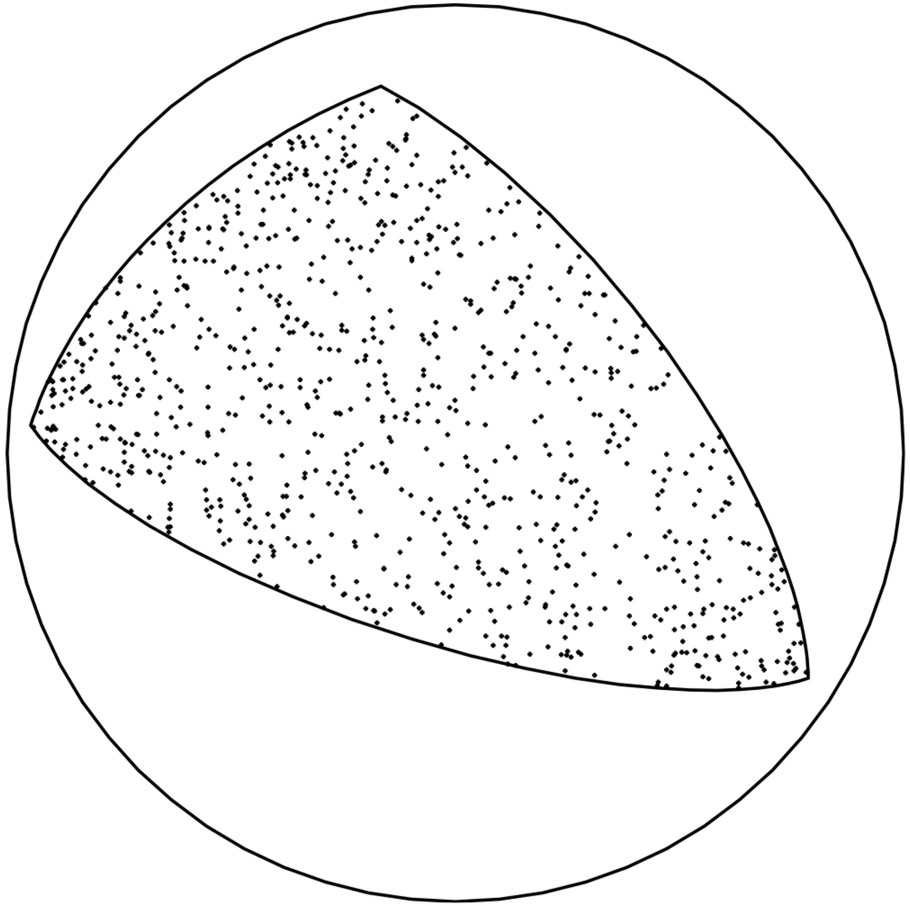
Generate cache samples for source values f **inside** domain:

no random walks needed



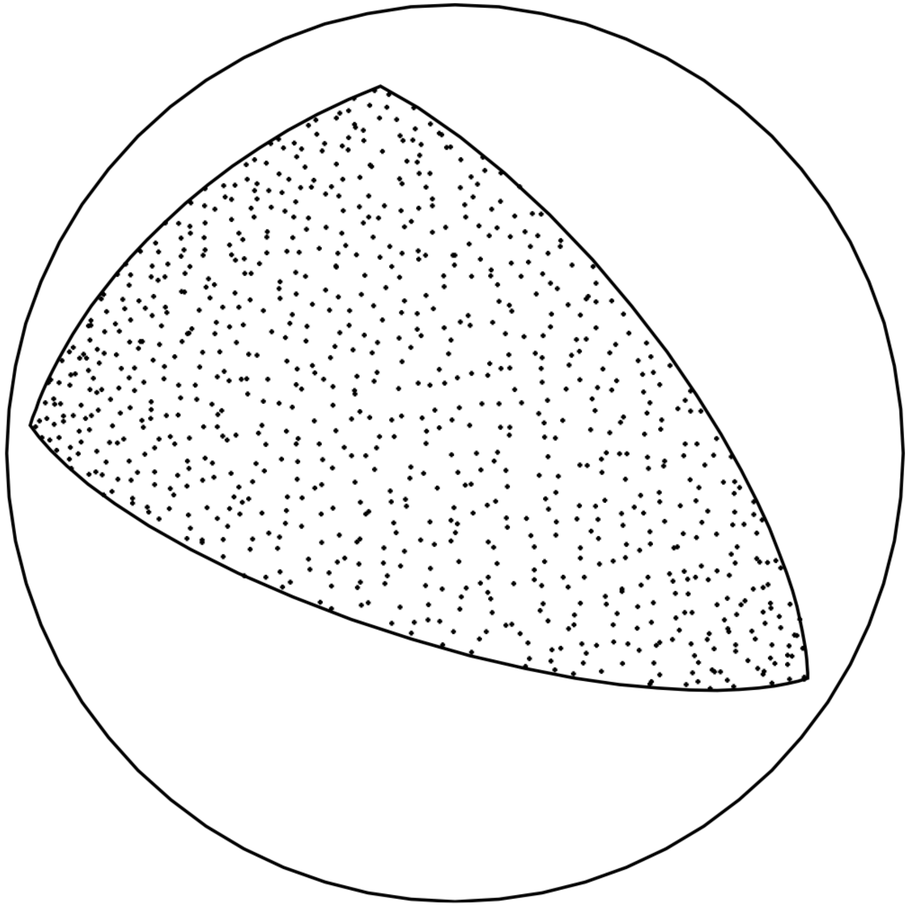
Stratification

uniform



"more clumpy"

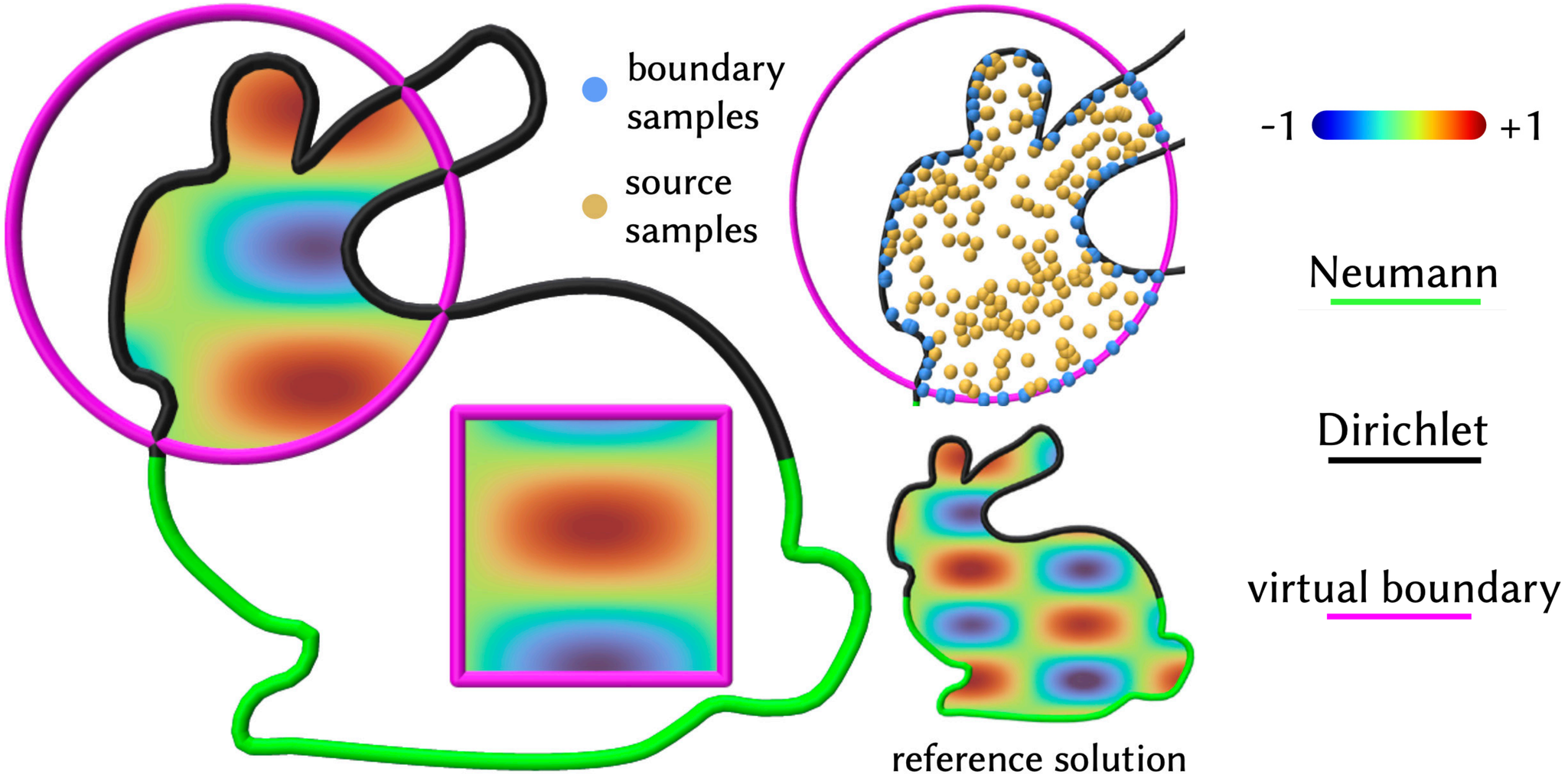
stratified



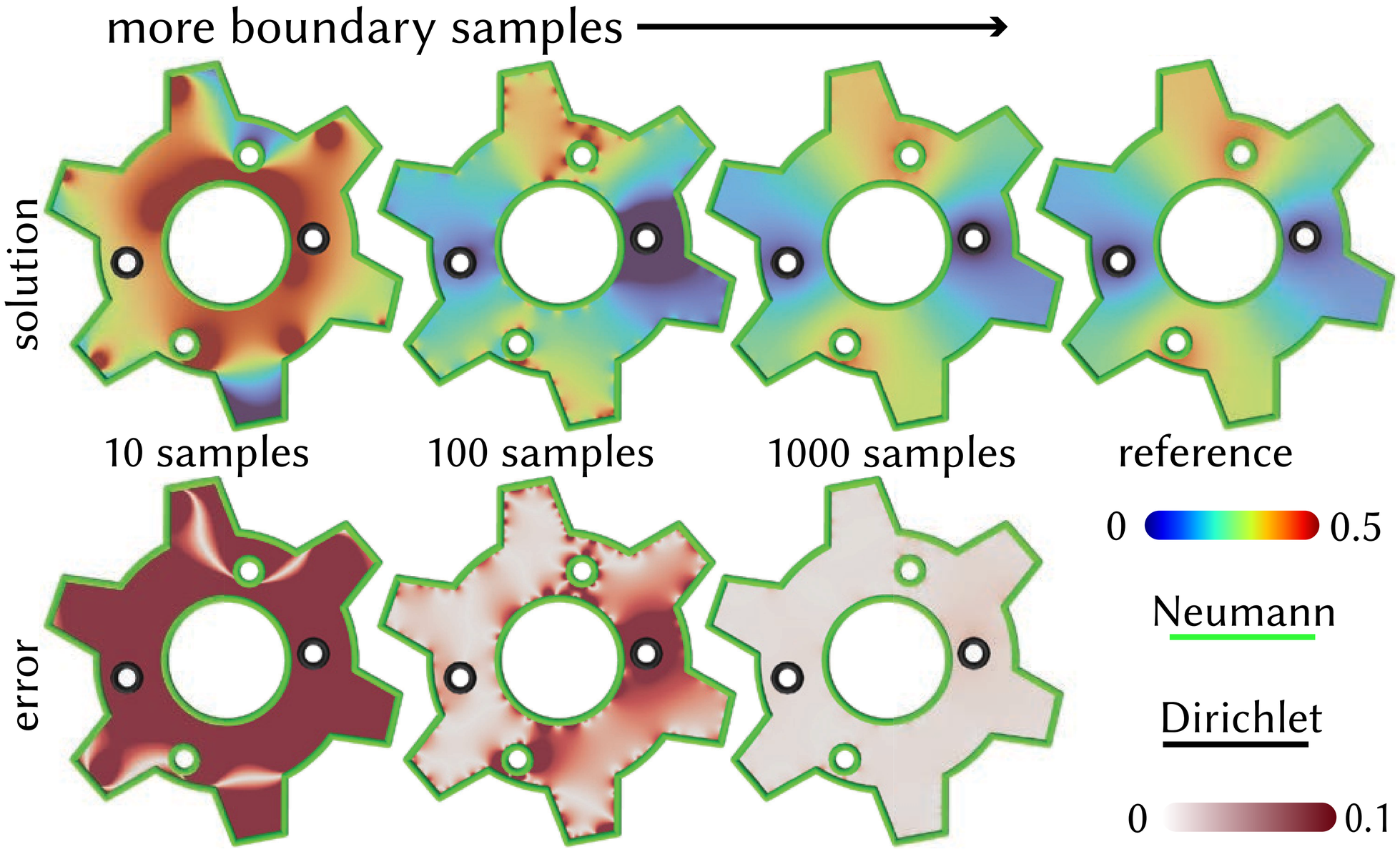
"more even"

Output sensitivity with BVC

Can focus computation in **local regions of interest**

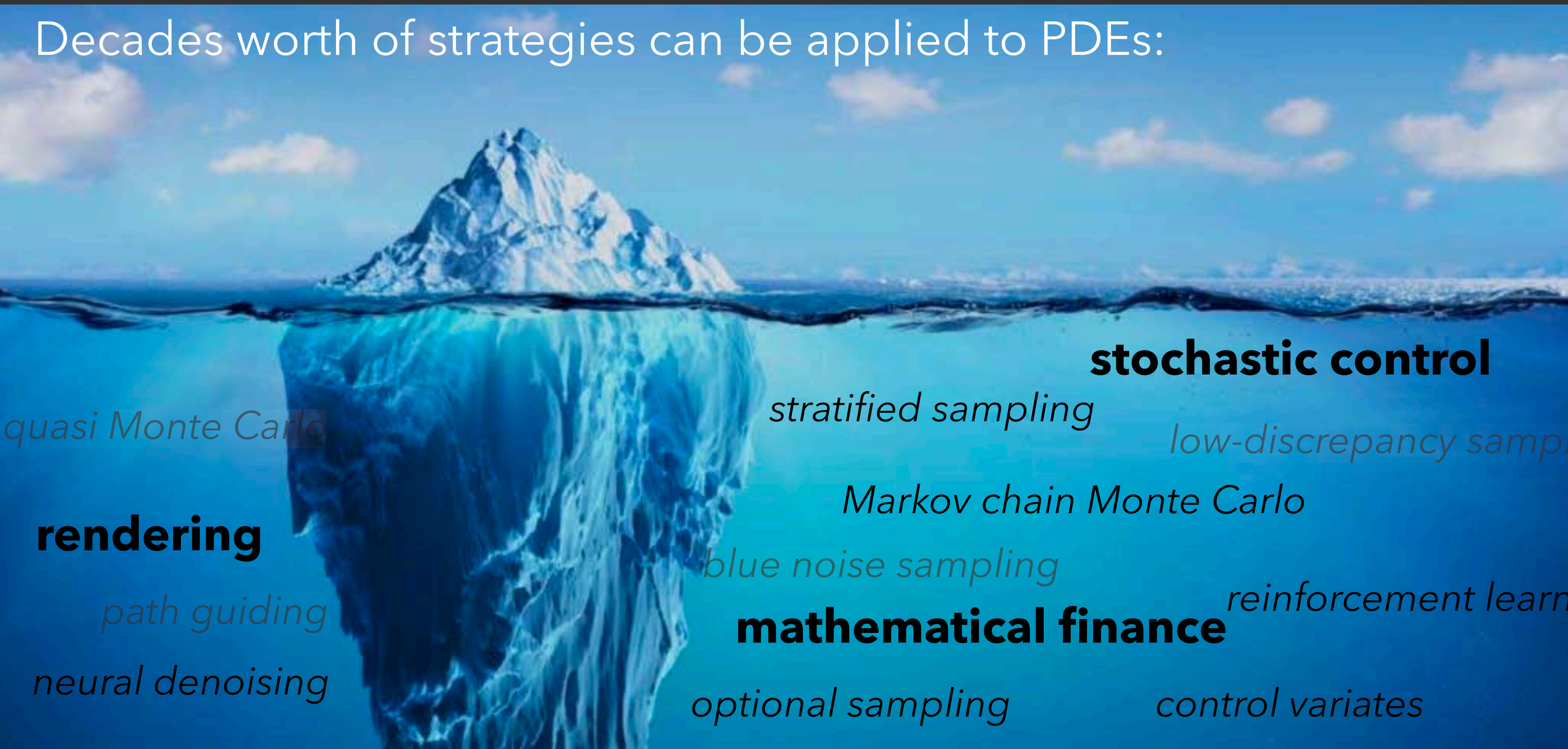


Error and convergence



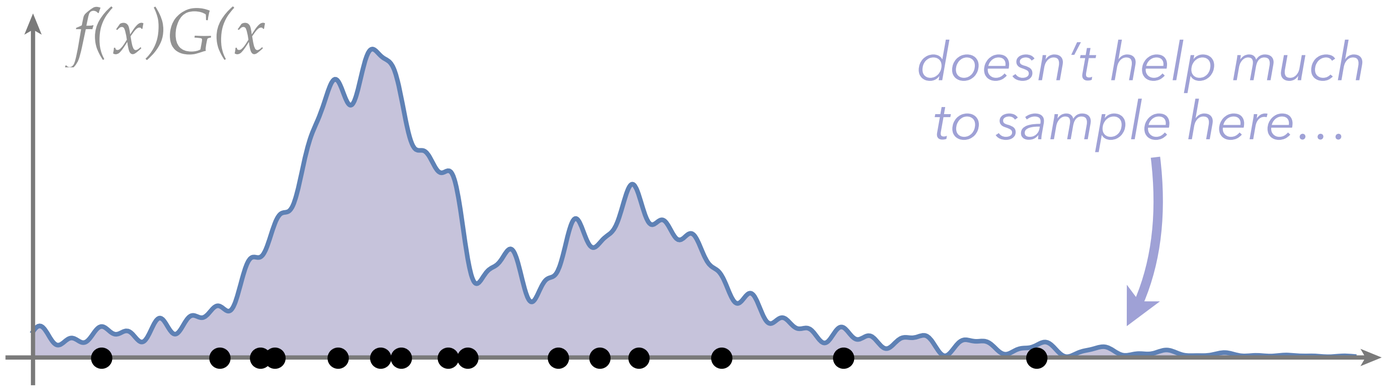
Just the tip of the iceberg…

Decades worth of strategies can be applied to PDEs:

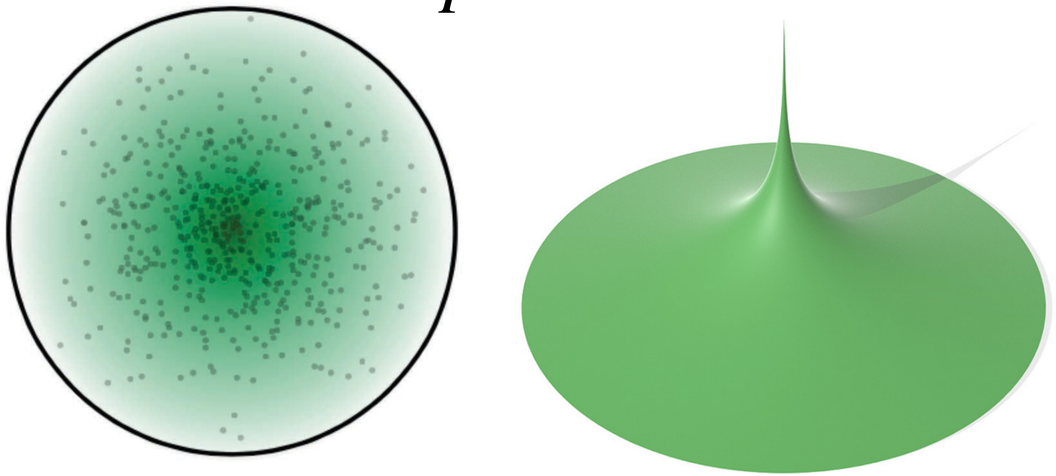


Importance sampling of Green's function & source term

$$\hat{I} = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)\ G(X_i)}{p(X_i)}$$

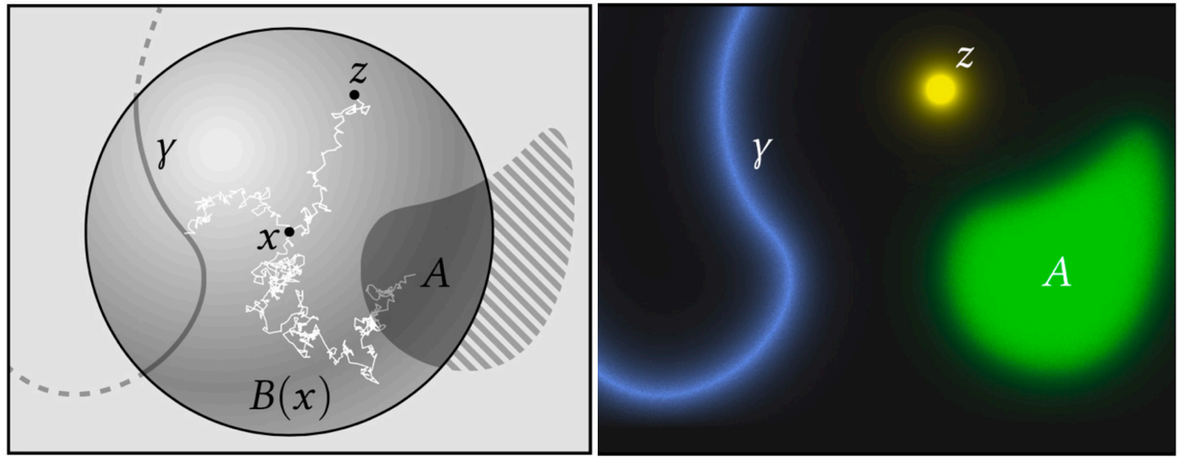


$p \propto G$



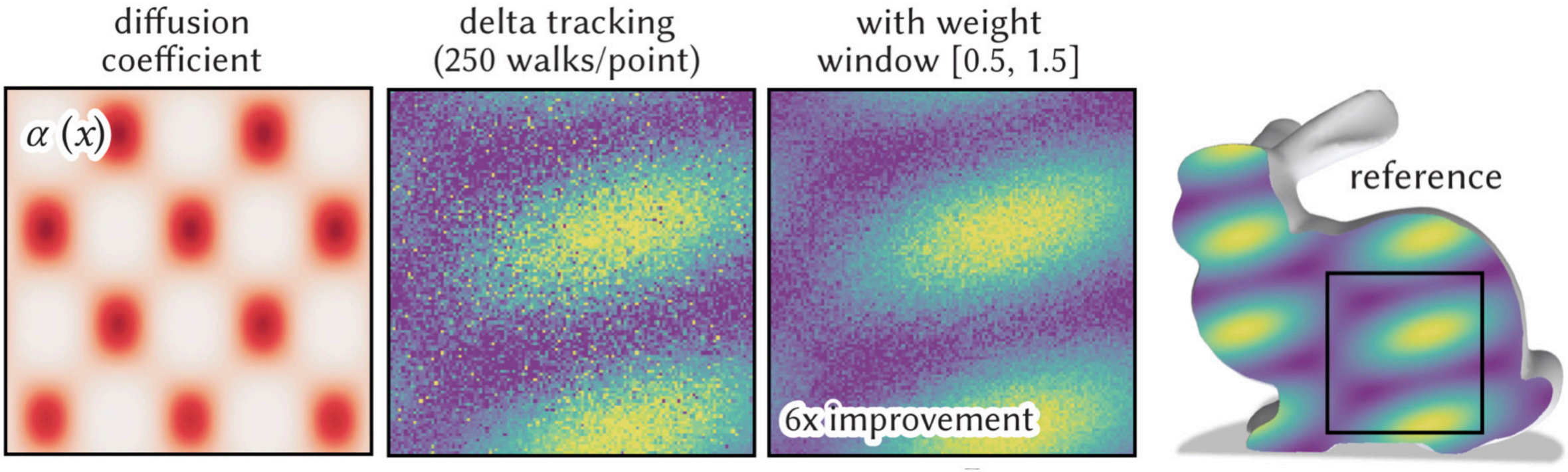
Samples drawn from Green's function of ball

$p \propto f$



Sampling point, curve & area sources

Weight window for variable coefficient PDEs



Probabilistically **terminate** low-contribution random walks to improve efficiency

Probabilistically **split** high-contribution random walks with for better exploration

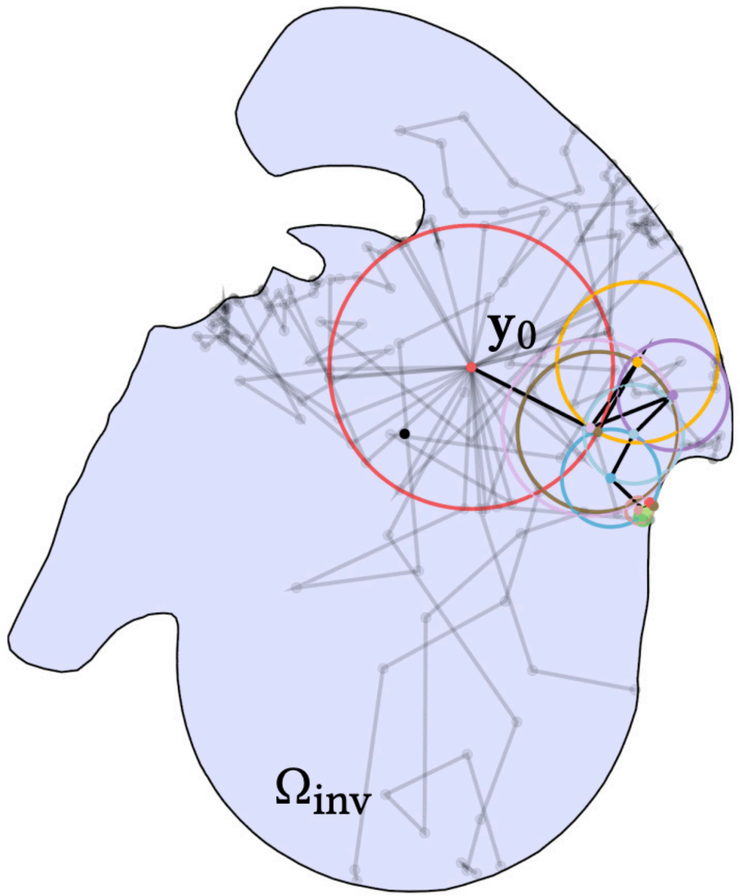
Sawhney et al., *"Grid-free Monte Carlo for PDEs with spatially varying coefficients"*, SIGGRAPH 2022

Kelvin transform for exterior problems

exterior problem

Invert via
Kelvin transform

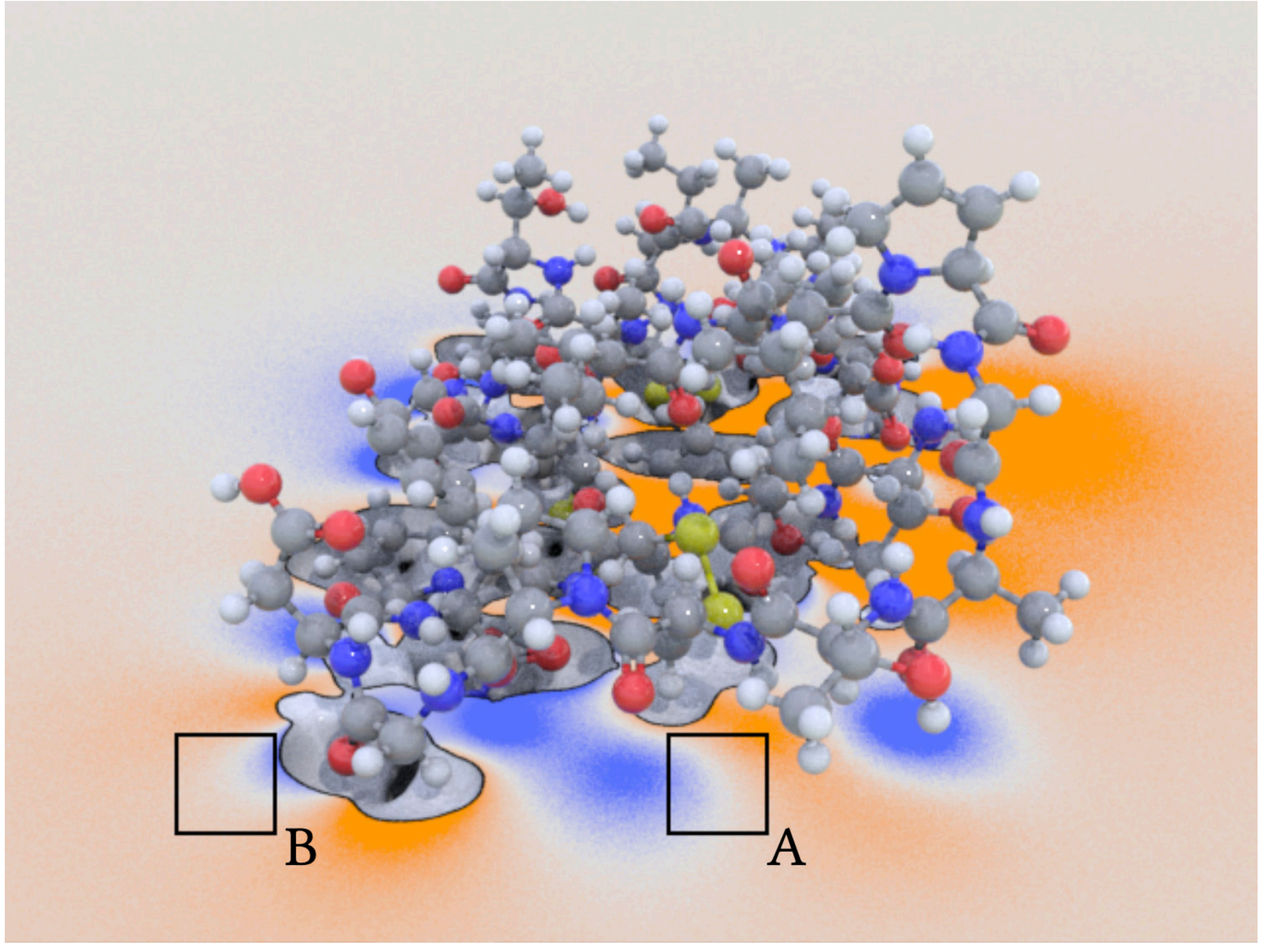
interior problem



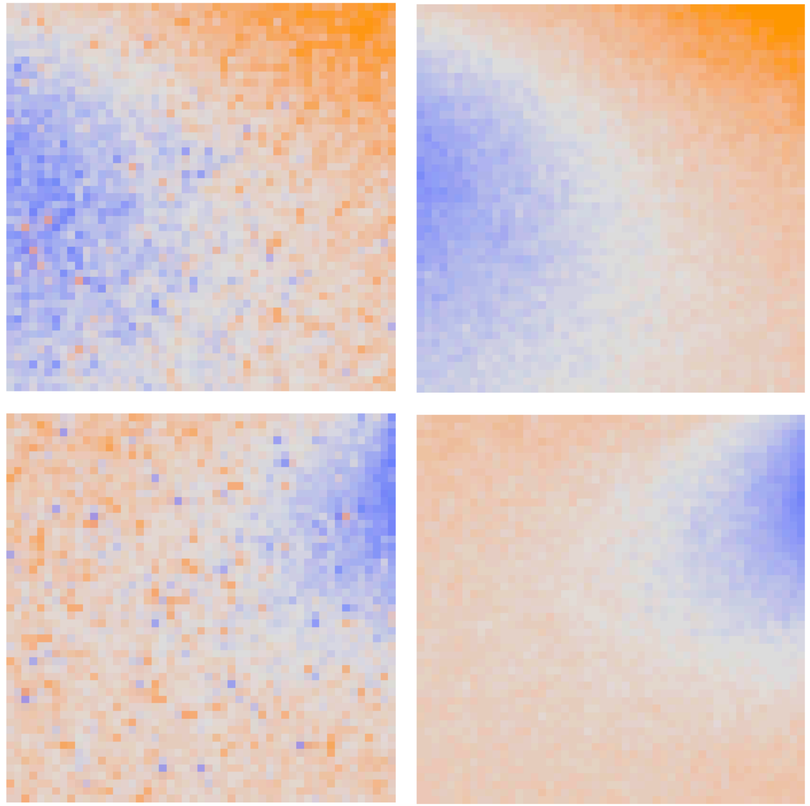
Nabizadeh et al., *"Kelvin Transformations for Simulations on Infinite Domains,"* SIGGRAPH 2021

Kelvin transform for exterior problems

Application: force evaluation for molecular dynamics simulation



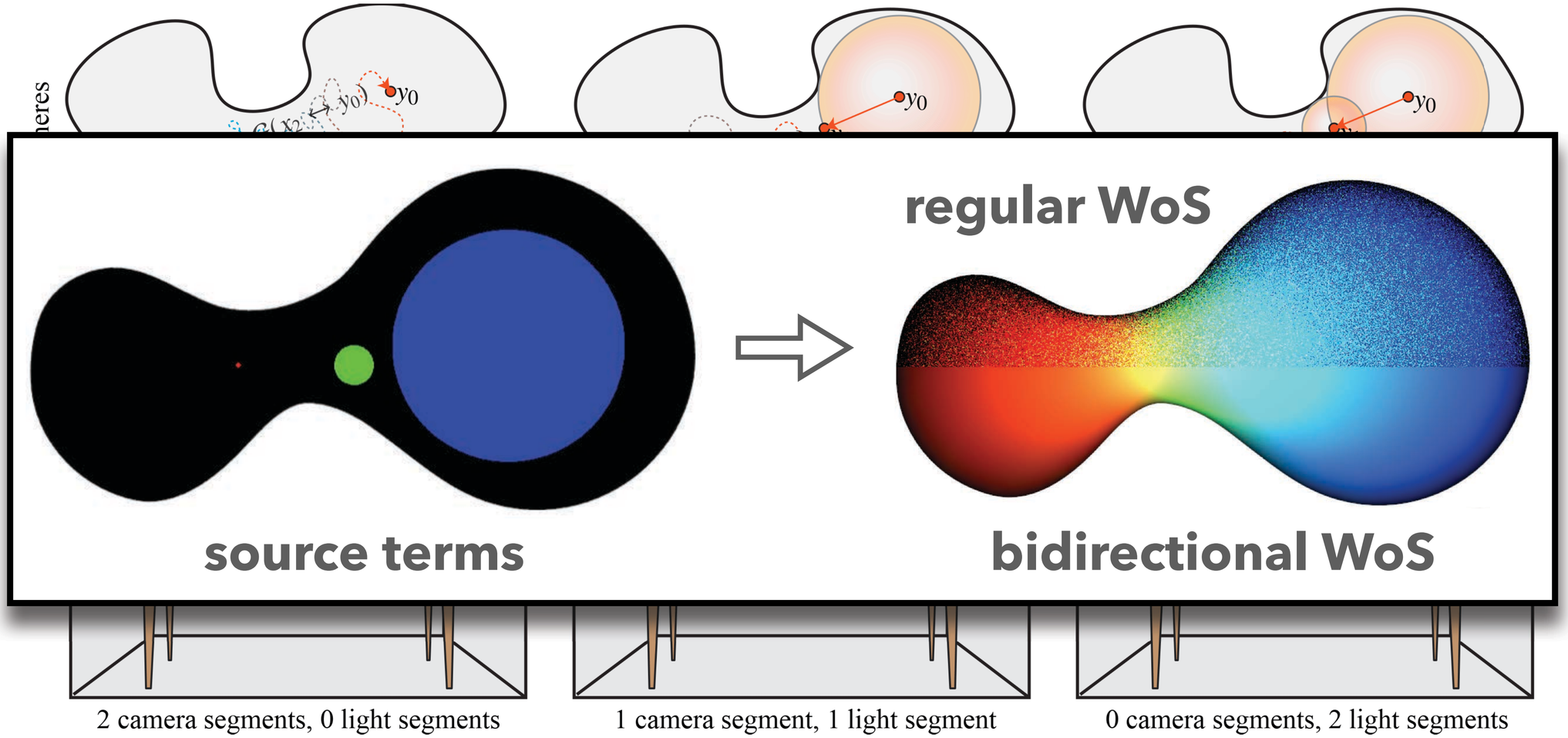
(electrostatic potential on 1CRN protein)



Russian Roulette **Kelvin Transform**

Bidirectional walk on spheres

Key idea: start at source points, estimate *Green's function* at sensor points

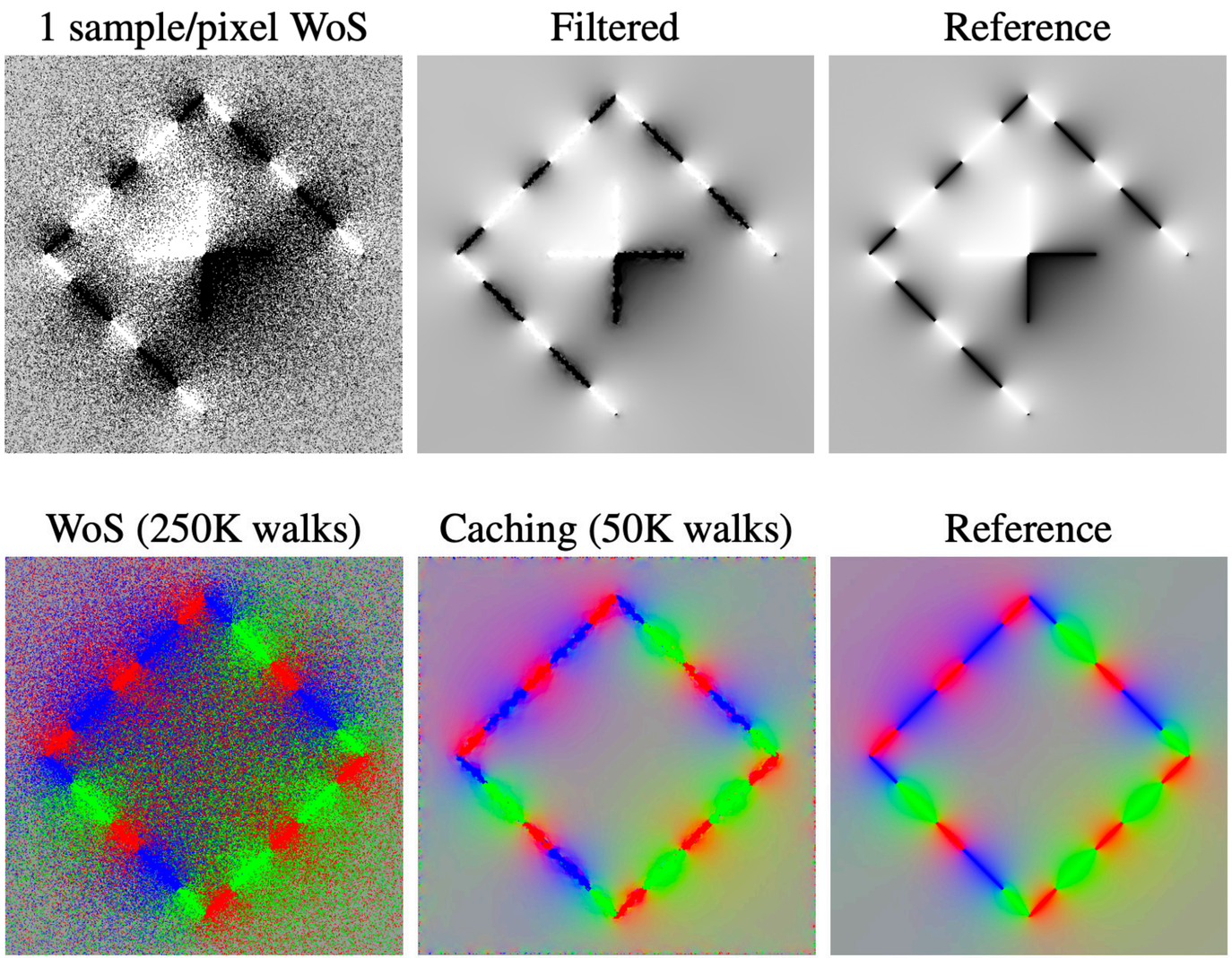


Qi, Seyb, Bitterli, Jarosz, *"A Bidirectional Formulation for Walk on Spheres,"* EGSR 2022

Sample reuse via Mean Value Caching

volumetric mean value property

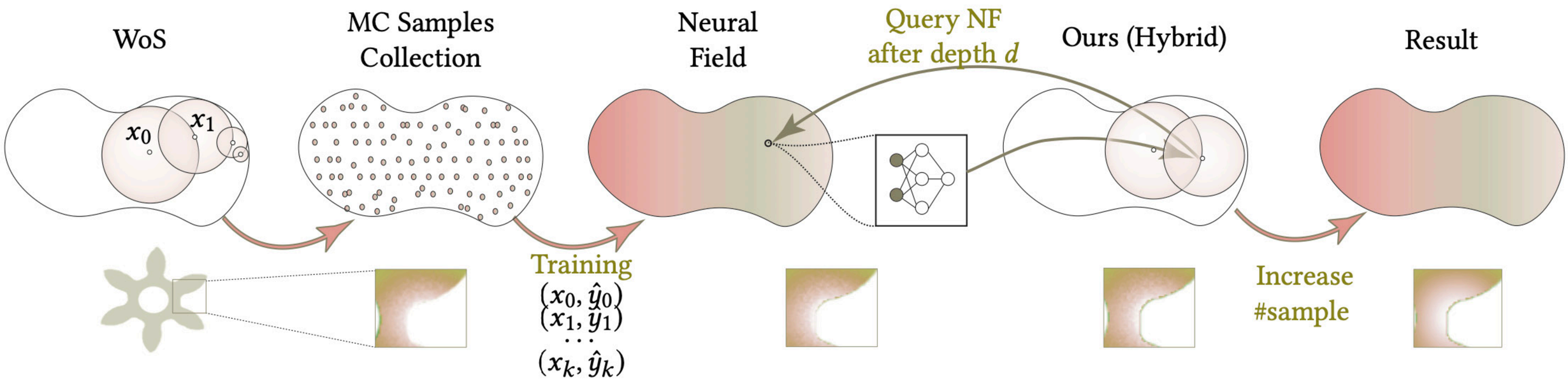
$$u(x) = \frac{1}{|B(x)|} \int_{B(x)} u(y)\ dy$$



Bakbouk & Peers, *"Mean Value Caching for Walk on Spheres,"* EGSR 2023

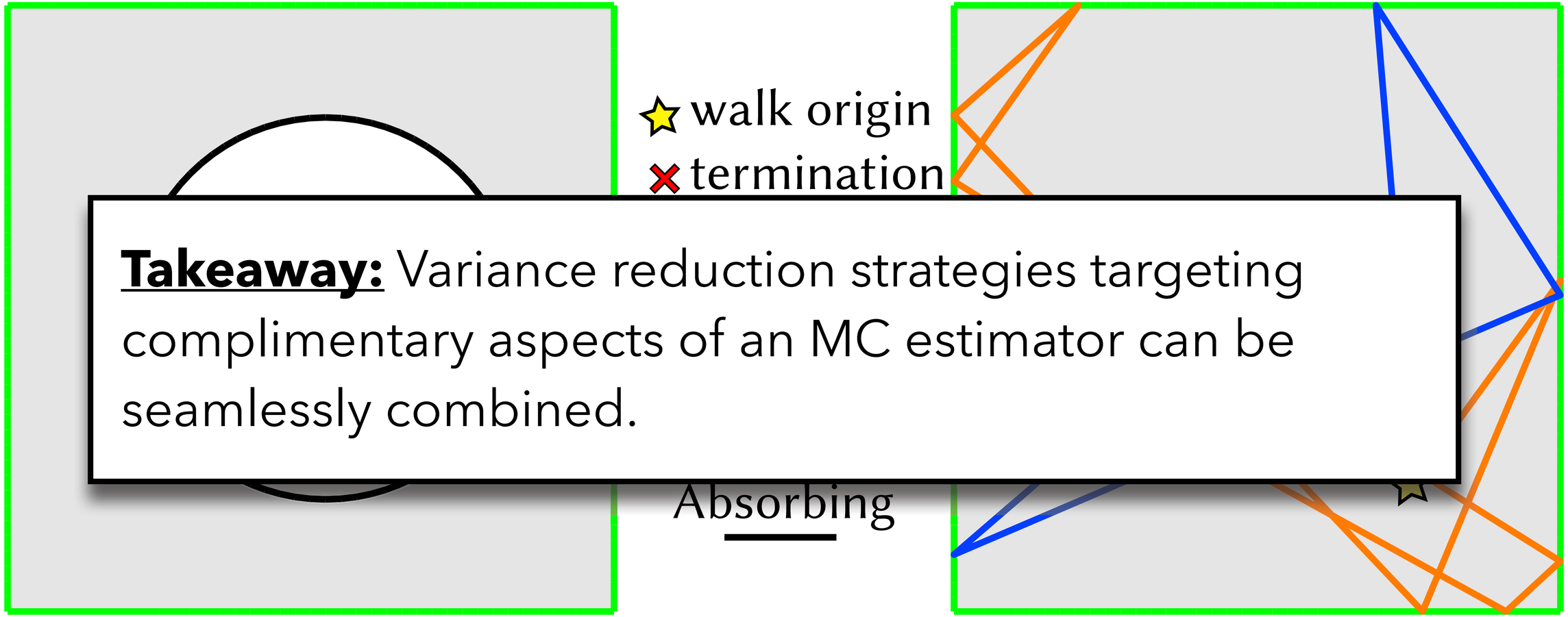
Neural Caches

Key idea: Terminate walks early by training network to predict solution



Li, Yang, Deng, De Sa, Hariharan, Marschner,
"Neural Caches for Monte Carlo Partial Differential Equation Solver," SIGGRAPH Asia 2023

Long walk lengths



Takeaway: Variance reduction strategies targeting complimentary aspects of an MC estimator can be seamlessly combined.

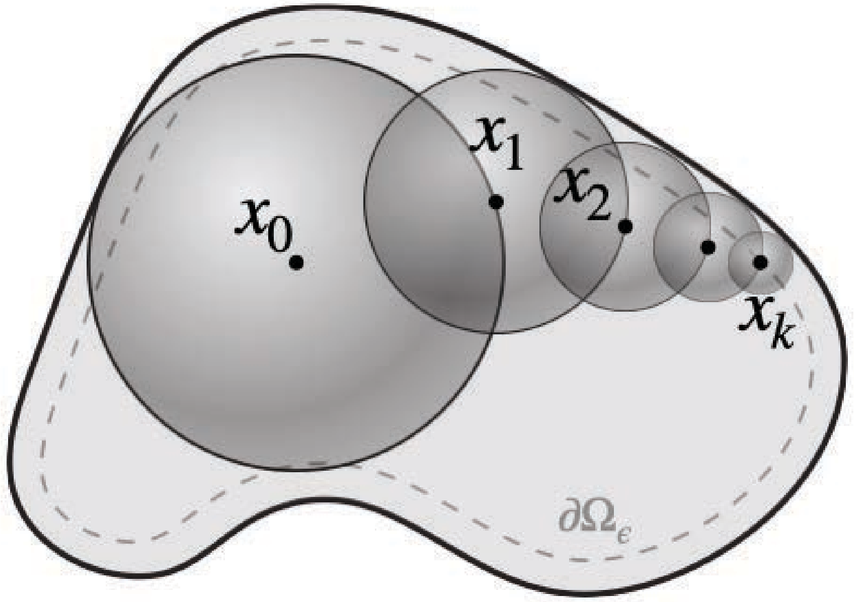
Ray Tracing Avg # Steps: 240

WoSt Avg # Steps: 219

PART6:
Evaluation, Recent & Future work

Summary

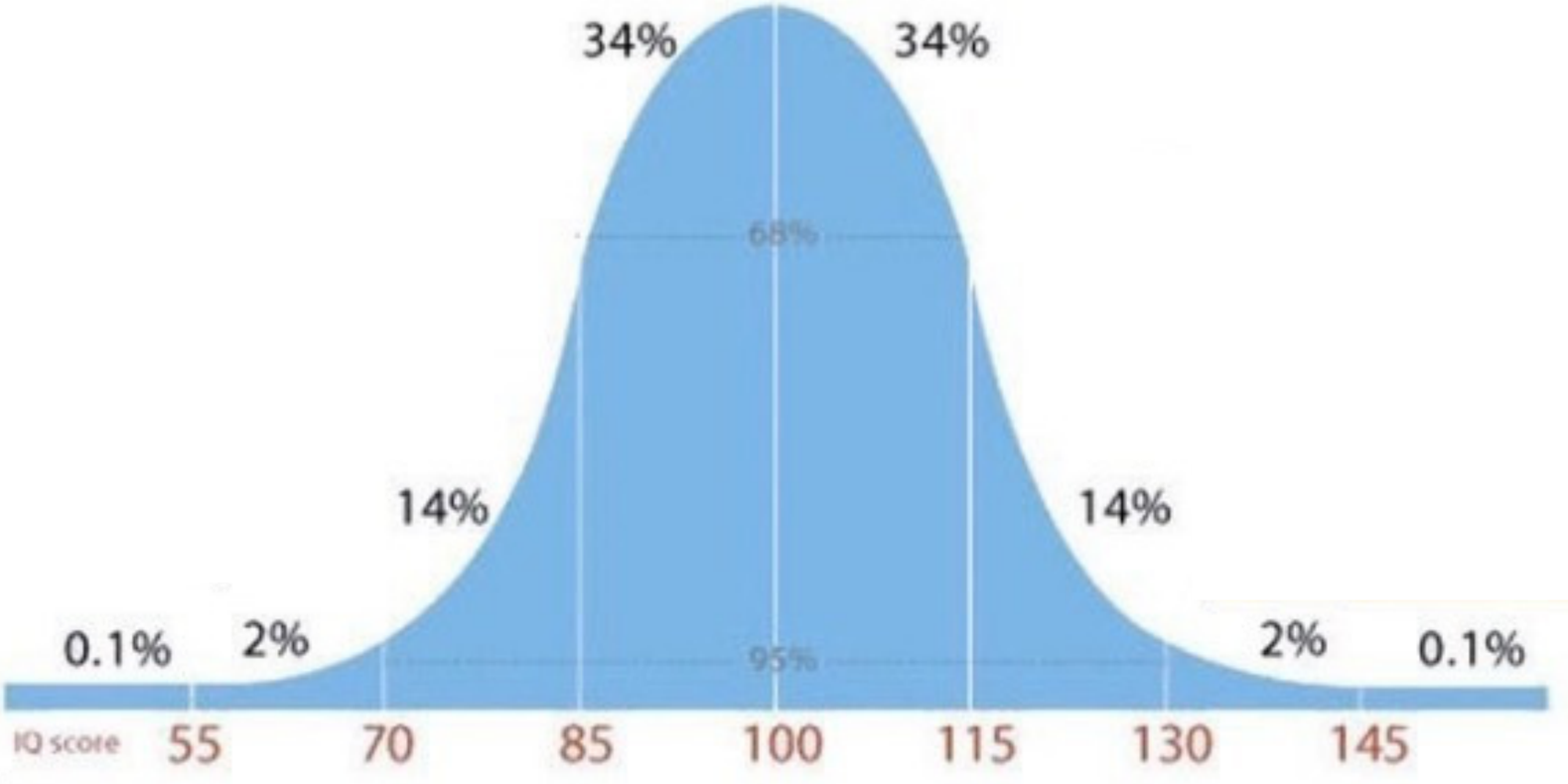
Presented emerging class of **grid-free MC methods** for PDEs based on WoS



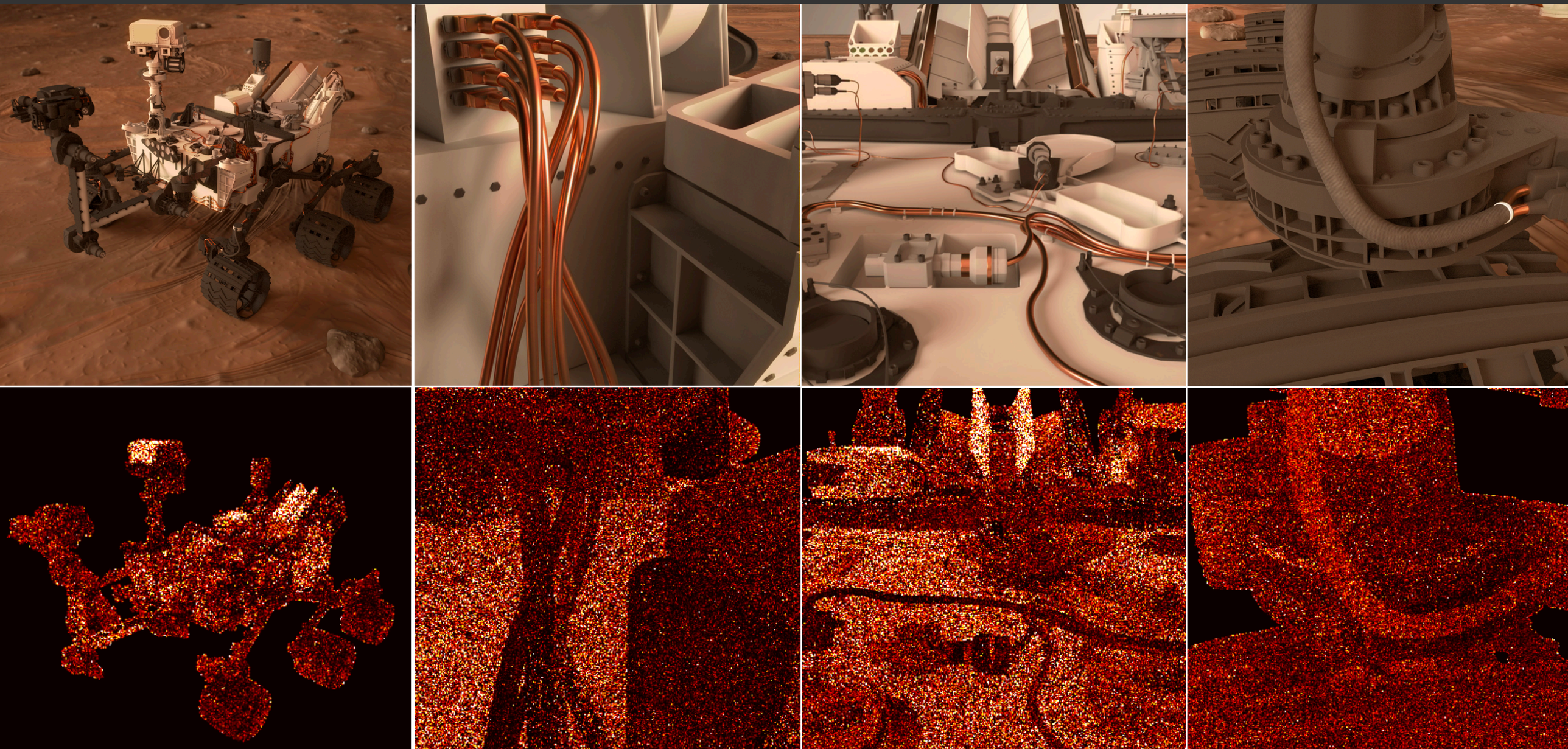
WoS reframes PDE as **integration problem**, and uses random sampling

WoS inherits **many advantages of Monte Carlo rendering** (scalable with geometric complexity, parallelism, output sensitive, …)

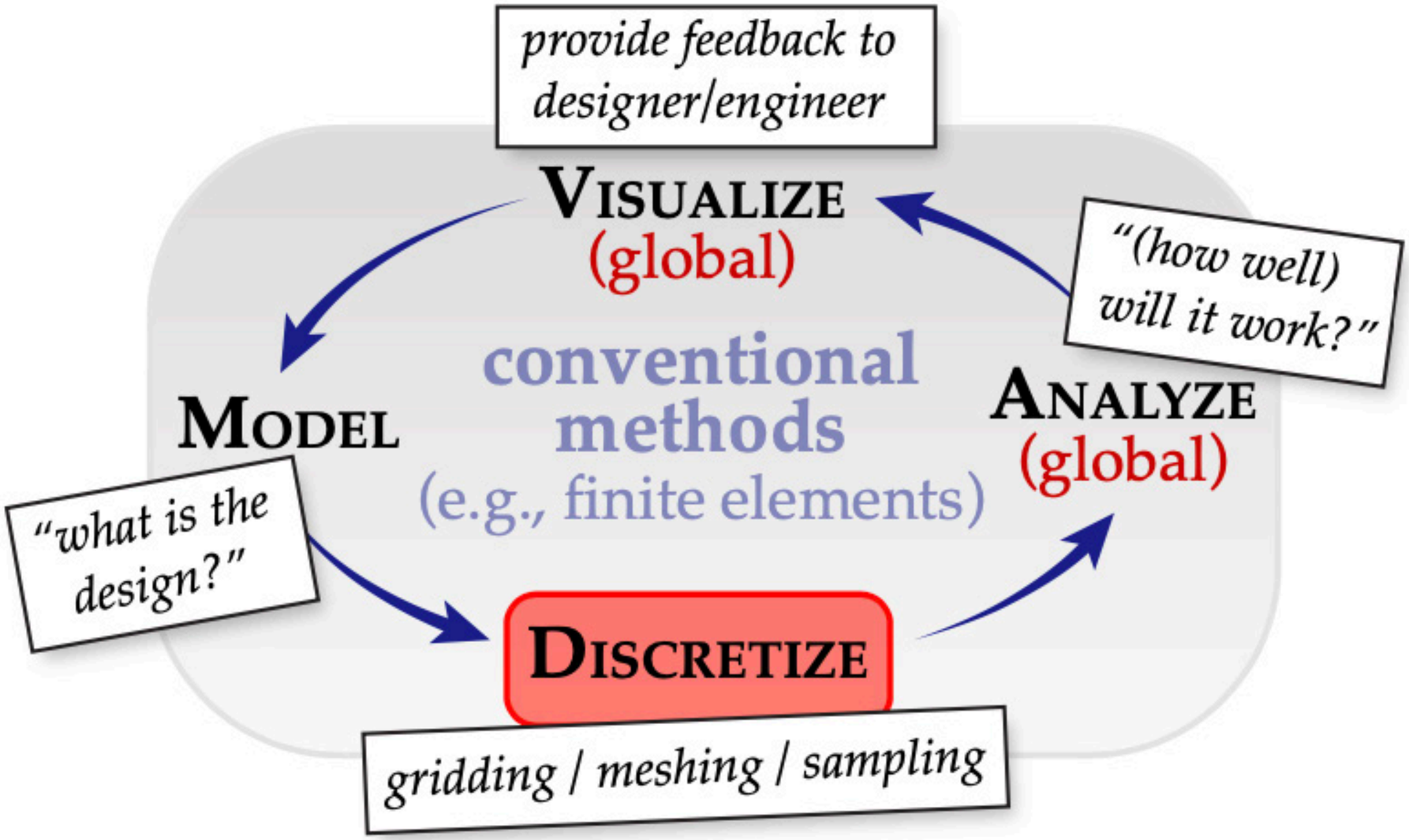
Not always the right tool for the job!



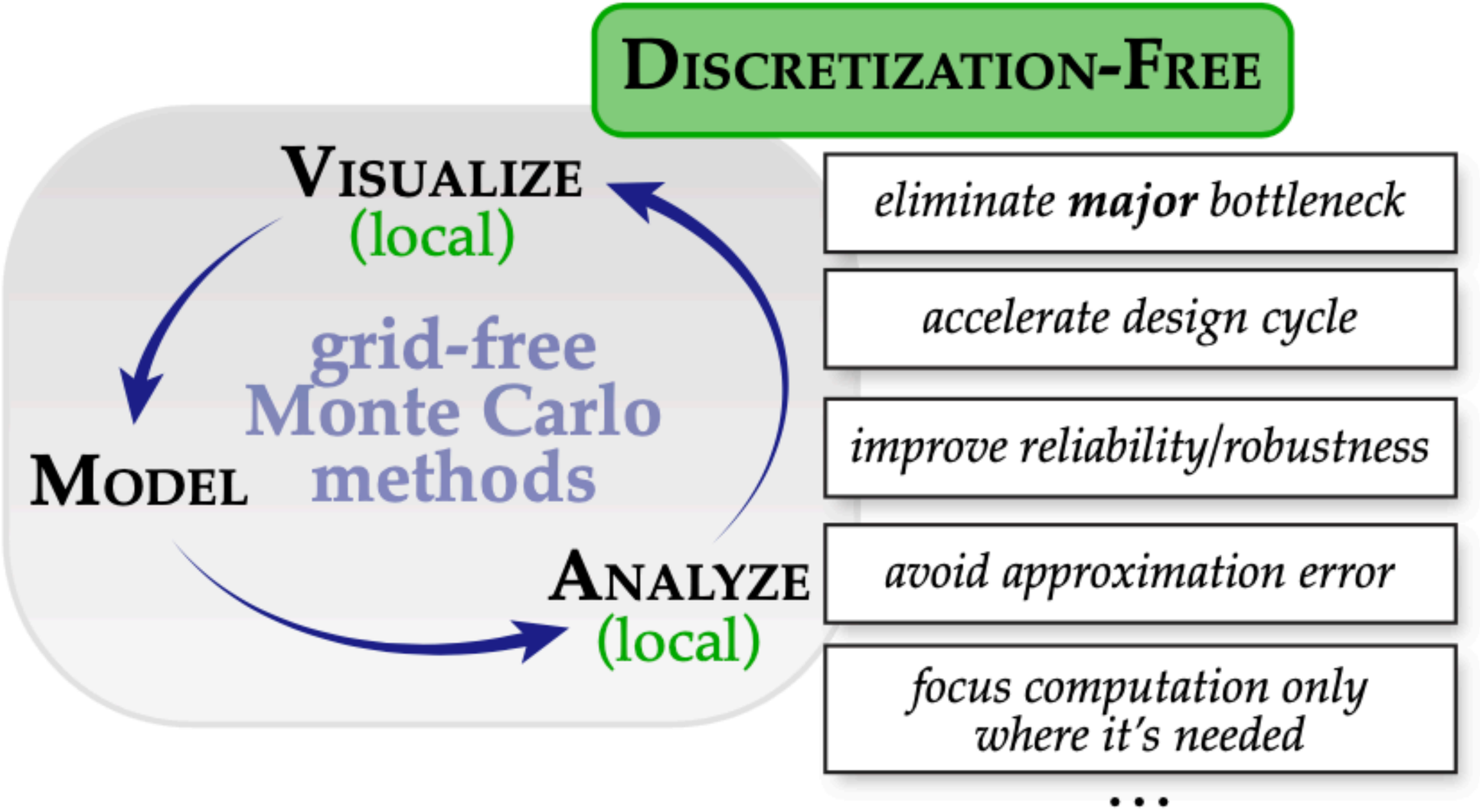
Monte Carlo is dumb, but it just works!



Traditional approach to PDE-based analysis

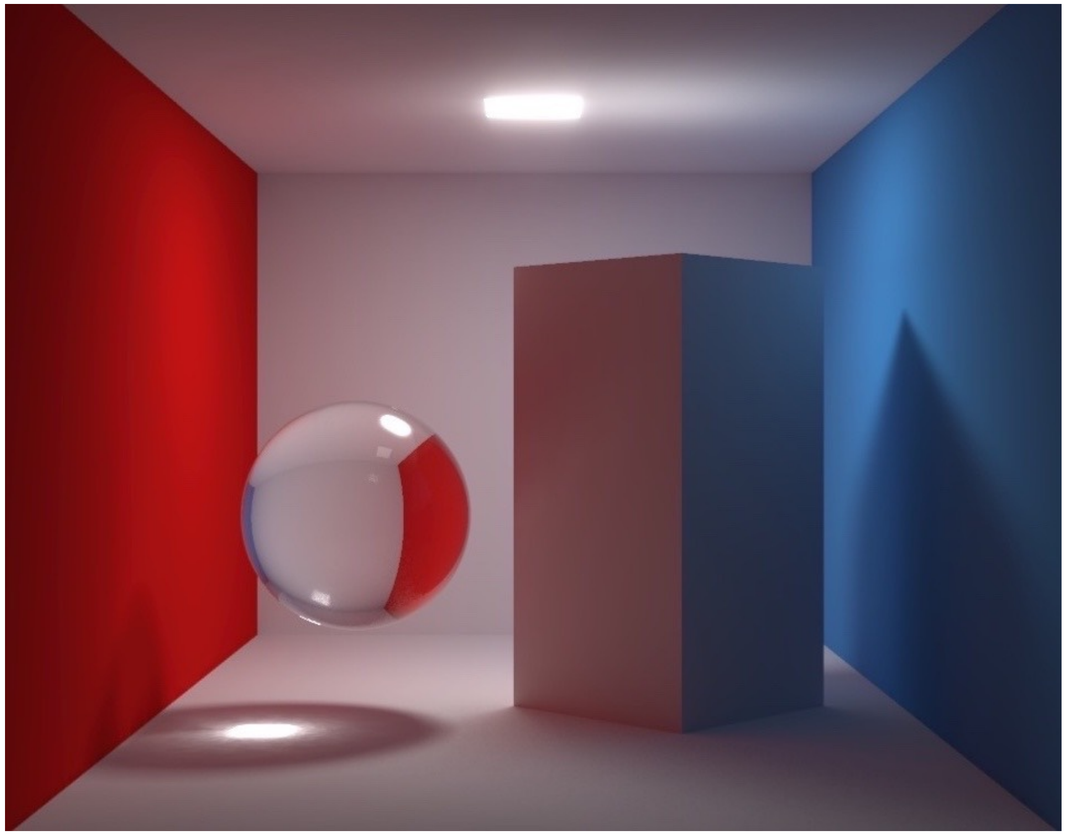


Monte Carlo approach to PDE-based analysis



Comparison with conventional solvers

Monte Carlo Ray Tracing



https://www.cs.cmu.edu/afs/cs/project/classes-ph/860.96/pub/www/montecarlo.mail

```
From: cn1@irz301.inf.tu-dresden.de (Nguyen, D.C.)
Subject: What's wrong w/ Monte-Carlo methods?
To: globillum@imag.fr (Global Illumination List)
Date: Mon, 28 Oct 1996 15:50:34 +0200 (MESZ)

I often ask myself : Monte-Carlo ray-tracing, is this the way to do globillum in the
future? After reading a lot of papers aboud MC-methods, i still get confused w/ their
terminologies. I can't see any advantage of these methods over traditional methods
(radiosity), except the fact that meshing is not needed…
```

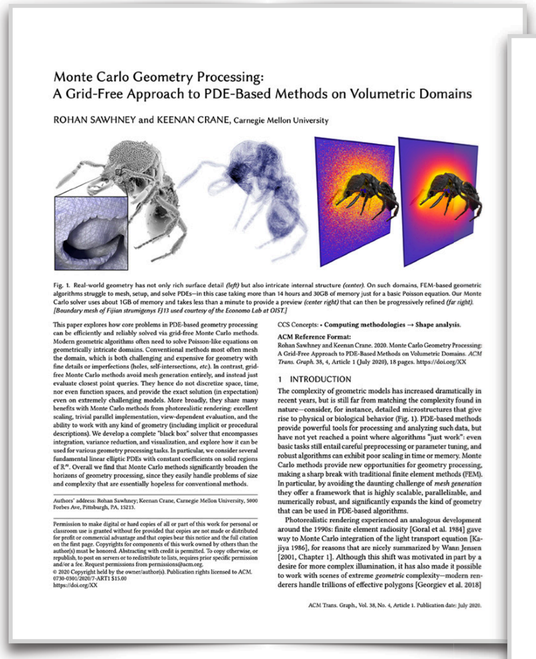
```
From: shirley@facility.cs.utah.edu
Subject: Re: What's wrong w/ Monte-Carlo methods?
To: cn1@irz301.inf.tu-dresden.de (Nguyen D.C.)
Date: Mon, 28 Oct 1996 08:31:22 -0700 (MST)
Cc: globillum@imag.fr, shirley@facility.cs.utah.edu (Peter Shirley)

…In summary, pure MCPT has only two advantages-- it is so dumb that it doesn't get
hit by big scenes, and it is easy to implement.
```

http://www.incompleteideas.net/IncIdeas/BitterLesson.html

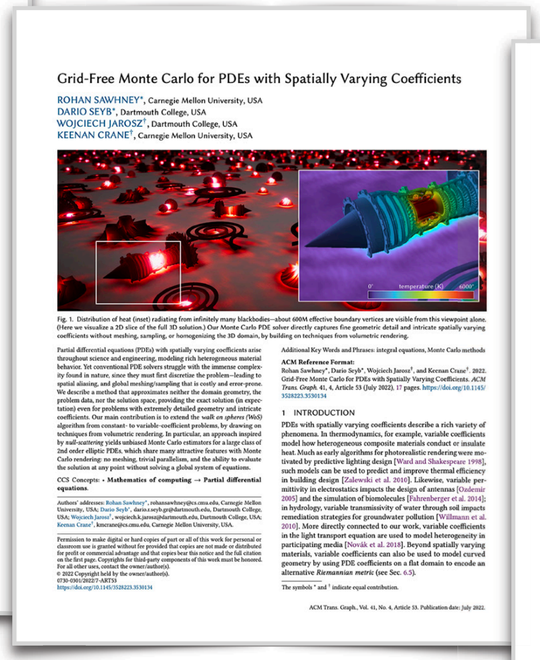
"One thing that should be learned from the bitter lesson is the great power of general purpose methods, of methods that continue to scale with increased computation even as the available computation becomes very great."

Monte Carlo Geometry Processing

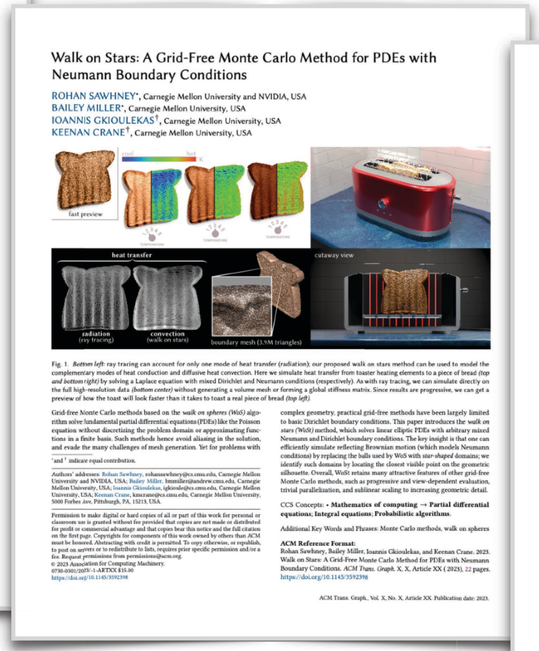
Introduce WoS to graphics & generalize it to solve larger class of PDEs



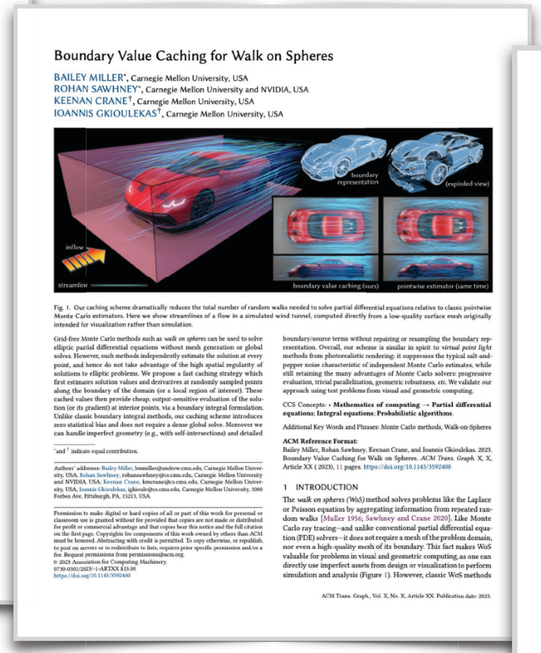
Sawhney & Crane
SIGGRAPH 2020



Sawhney*, Seyb*,
Jarosz†, Crane†
SIGGRAPH 2022



Sawhney*, Miller*,
Gkioulekas†, Crane†
SIGGRAPH 2023



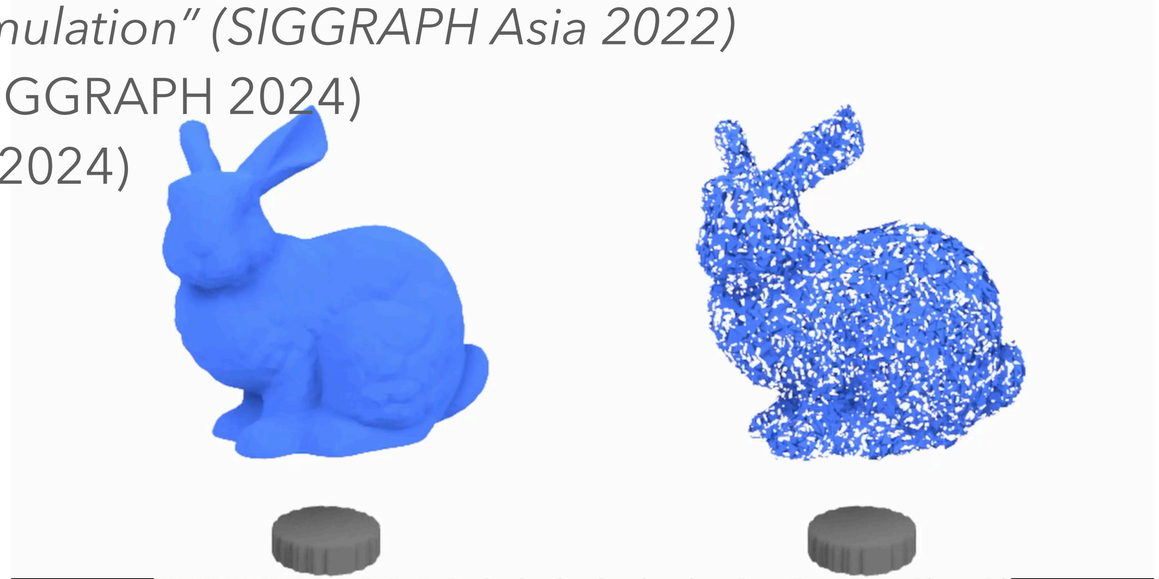
Miller*, Sawhney*,
Crane†, Gkioulekas†
SIGGRAPH 2023

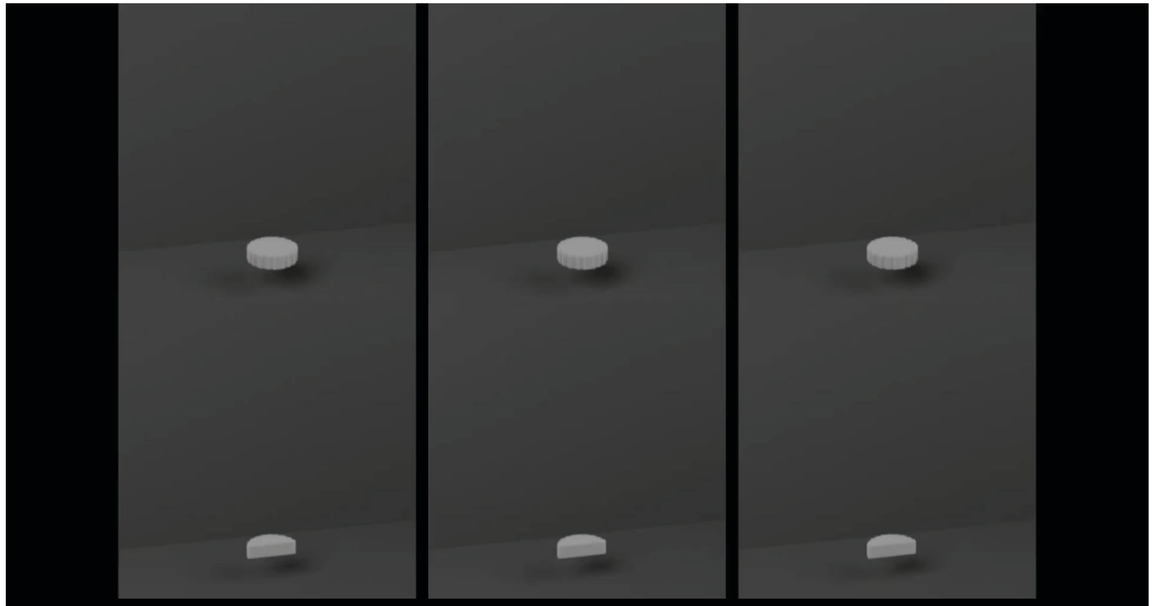
Miller*, Sawhney*,
Crane†, Gkioulekas†
SIGGRAPH 2024

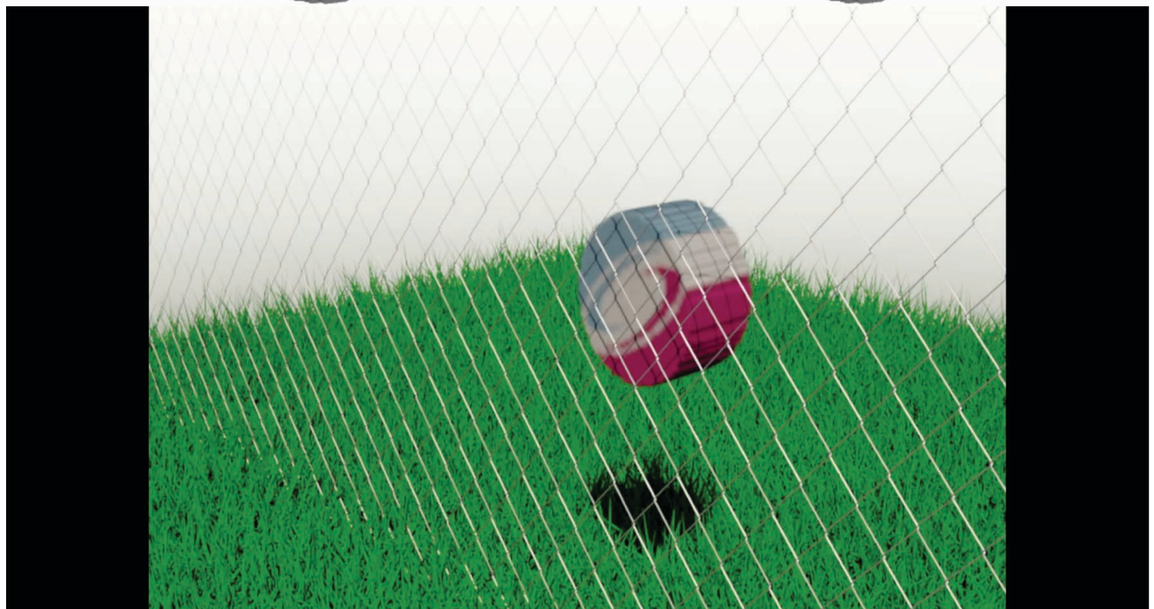
* and † denote equal contribution

Monte Carlo fluid simulation

Rioux-Lavoie et al, *"A Monte Carlo Method for Fluid Simulation" (SIGGRAPH Asia 2022)*
Sugimoto et al, *"Velocity based Monte Carlo Fluids"* (SIGGRAPH 2024)
Jain et al, "Neural Monte Carlo Fluid Sim" (SIGGRAPH 2024)

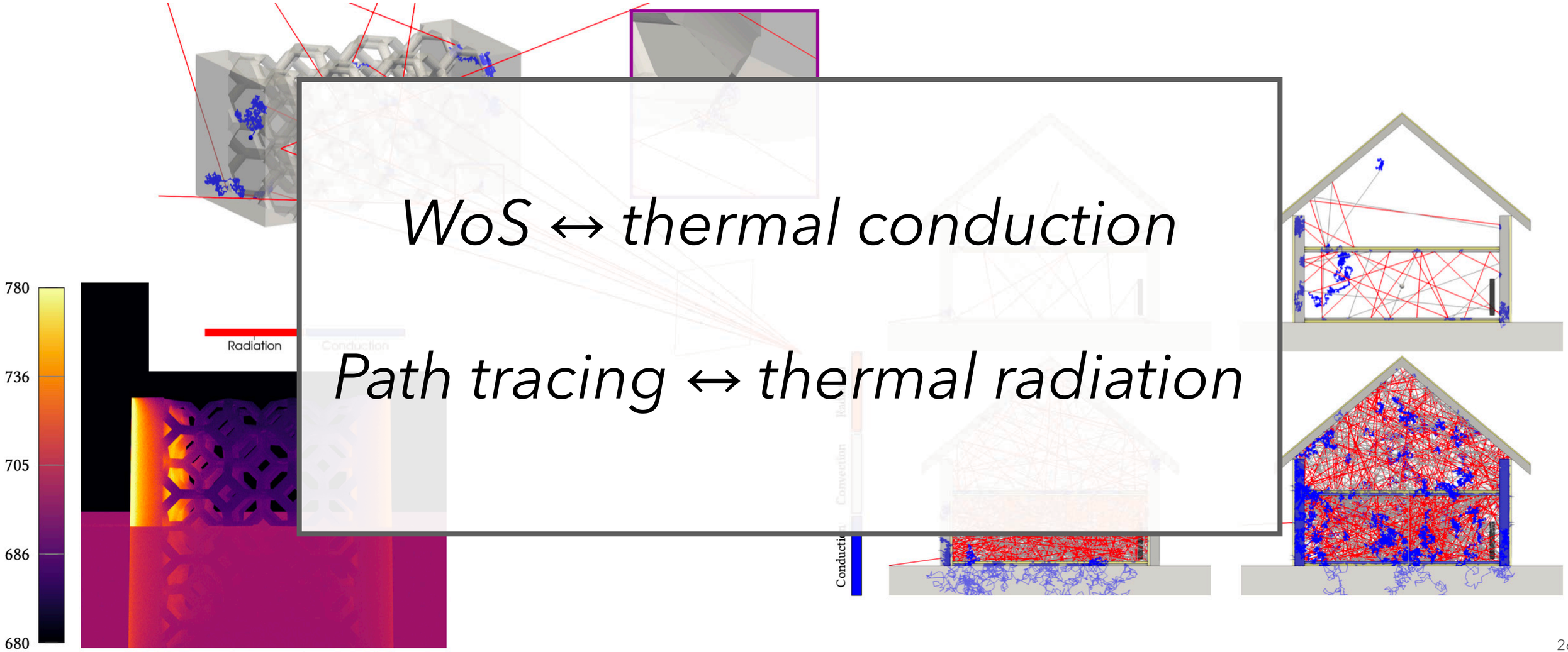






Monte Carlo Infrared Imaging

"Coupling Conduction, Convection and Radiative Transfer in a Single Path-Space: Application to Infrared Rendering"
Bati, Blanco, Coustet, Eymet, Forest, Fournier, Gautrais, Mellado, Paulin, Piaud, SIGGRAPH 2023

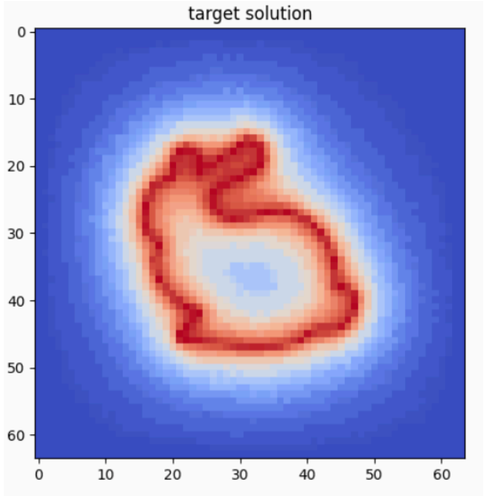


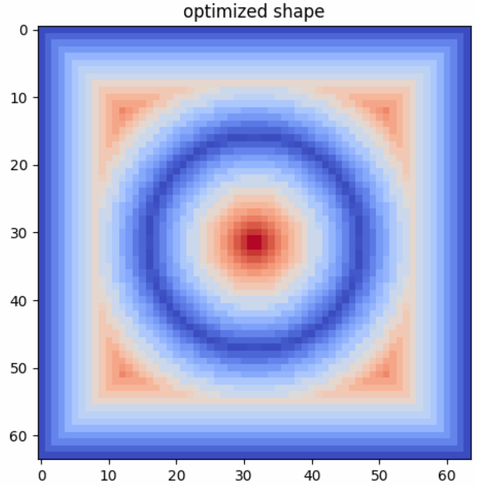
WoS ↔ thermal conduction

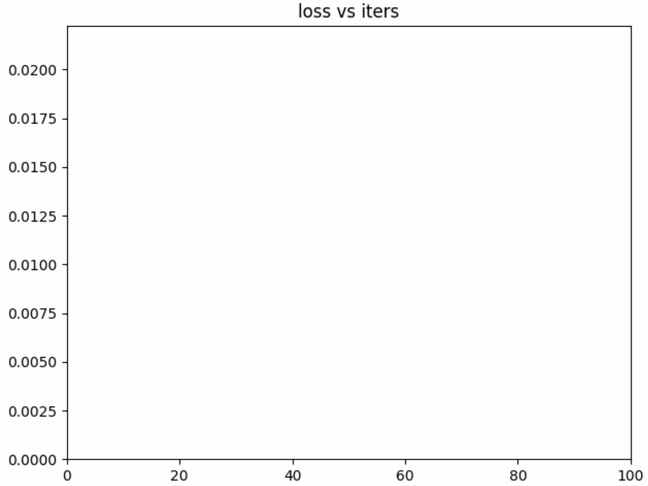
Path tracing ↔ thermal radiation

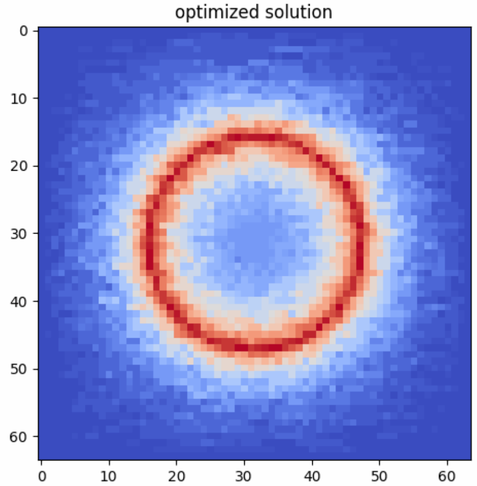
Differentiable solvers for PDE-based shape optimization

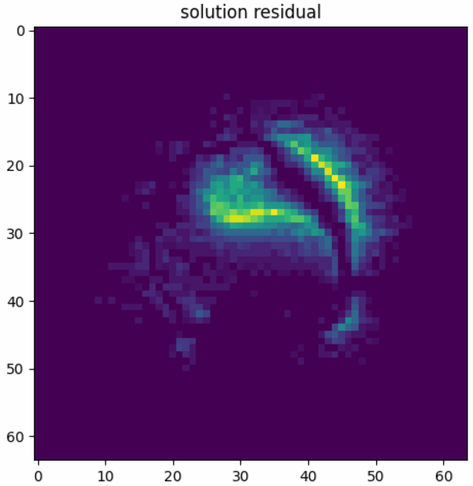
Goal: recover shape given measurements, e.g., temperature, electric potential



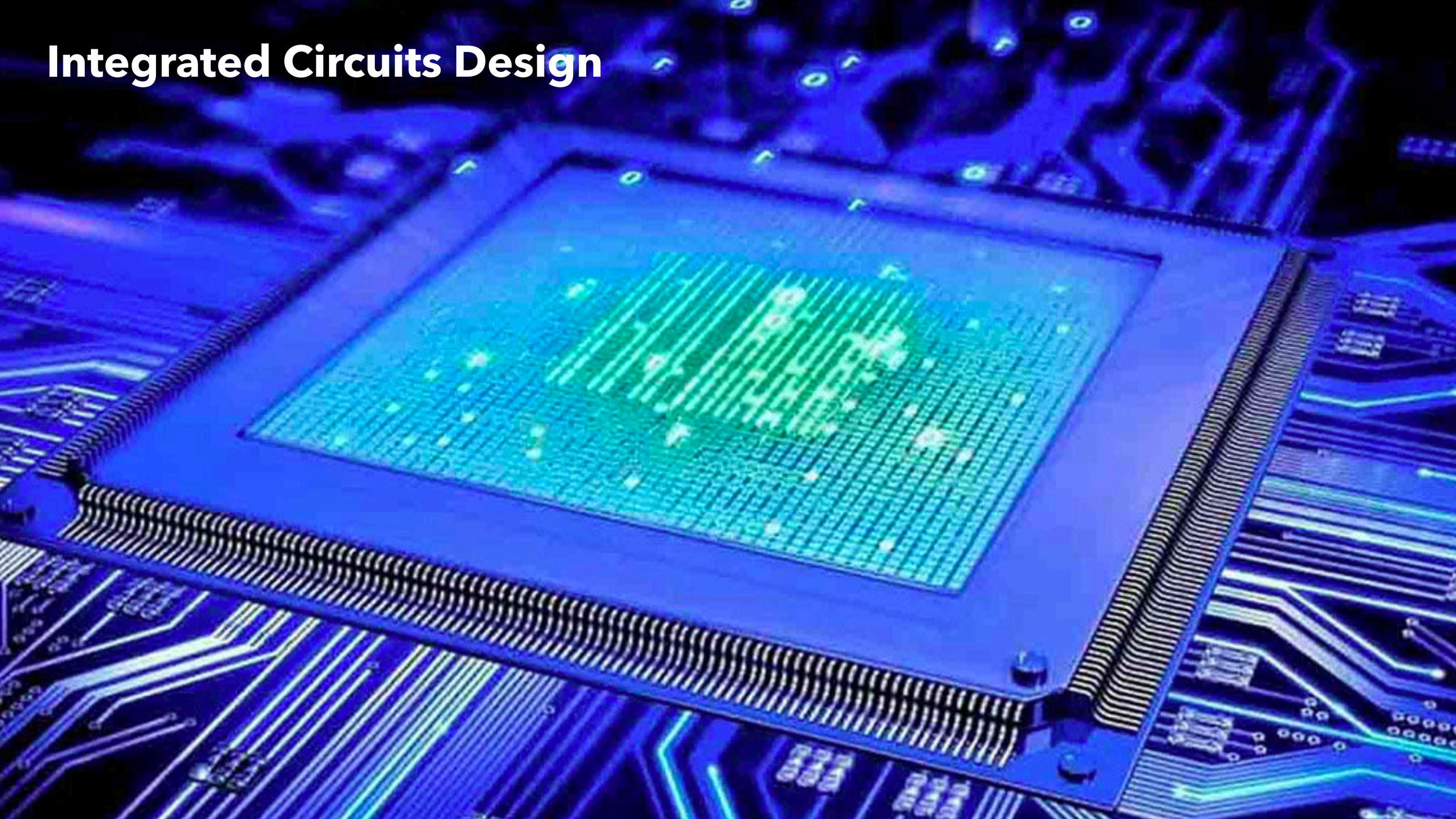




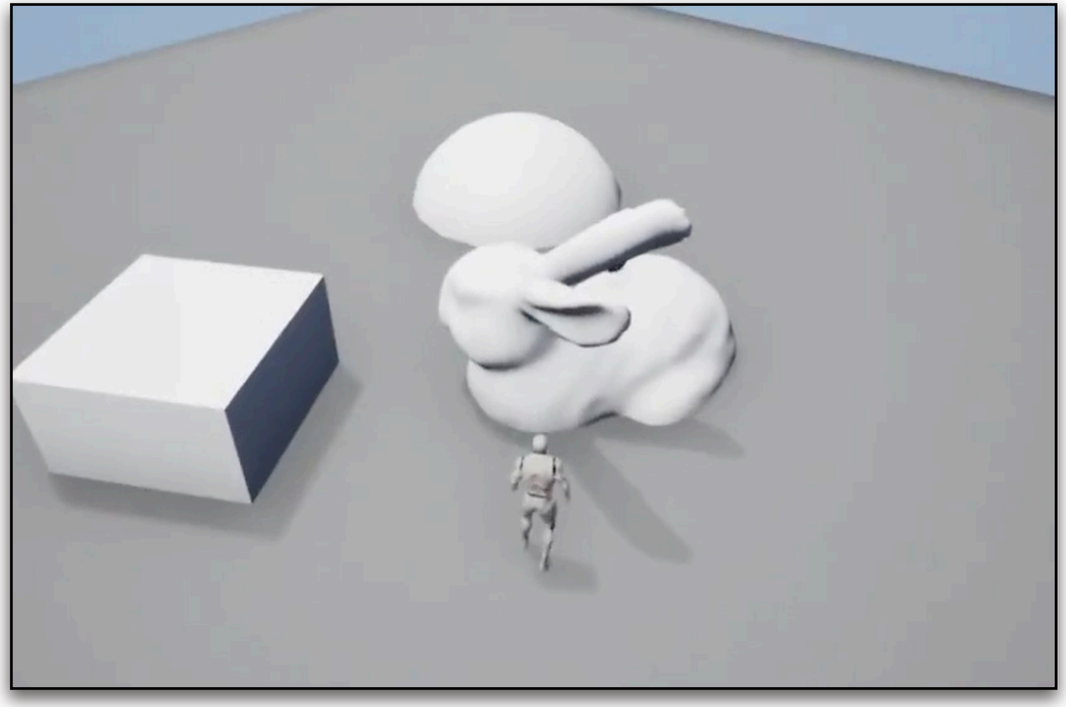


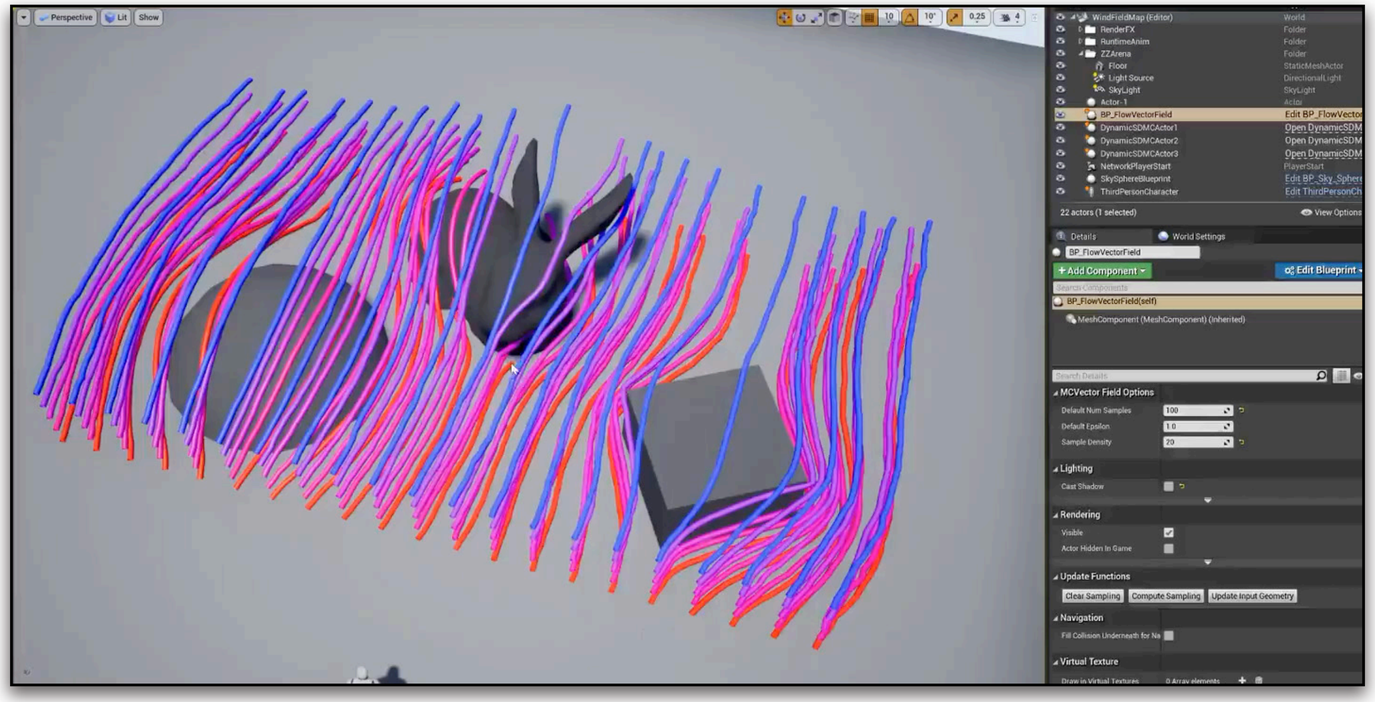


Integrated Circuits Design



Agent path planning





Credit: Ryan Schmidt

Next Steps

Extend WoS to more physical problems

– Heat, Helmholtz & wave equations, anisotropic diffusion, linear elasticity

– Multi-physics: couple conduction, radiation, convection

– Differentiability

Research → Production

– High-performance GPU implementation (**coming soon!**)

– Variance reduction: denoising & supersampling

"Real-world" applications

– exploit unique capabilities of Monte Carlo

Resources for further learning

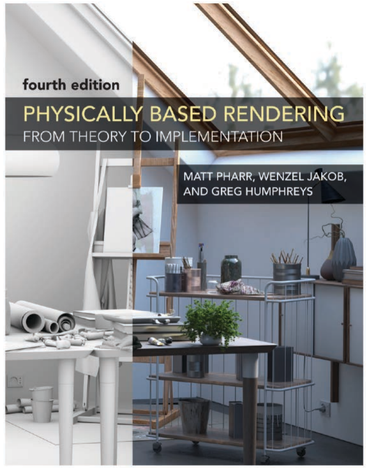
Slides are available online

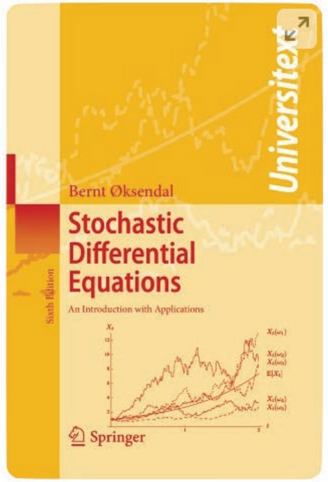
Rohan's PhD Thesis

Physically based rendering (PBRT)

Monte Carlo Methods and Applications (CMU)

Stochastic Differential Equations (Oksendal)





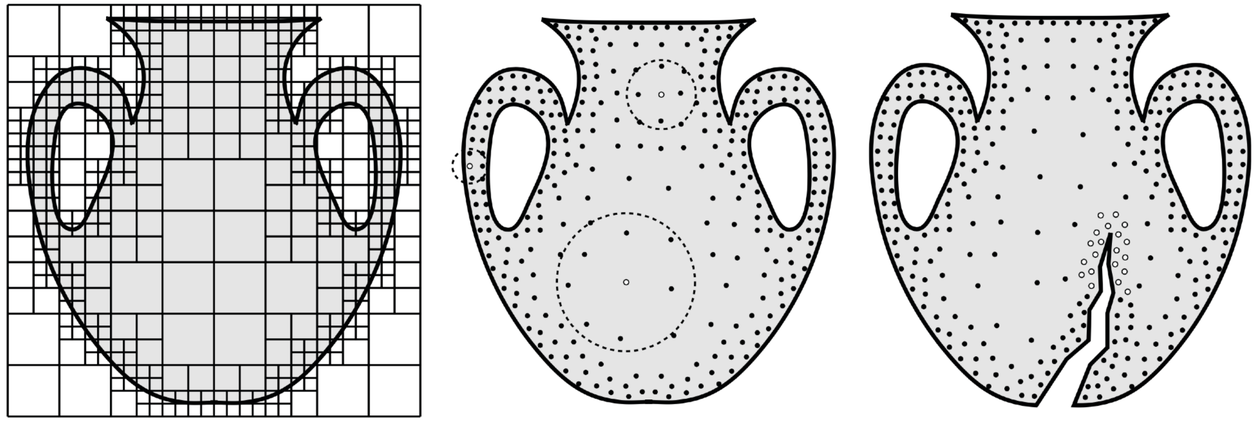
END.

BACKUP

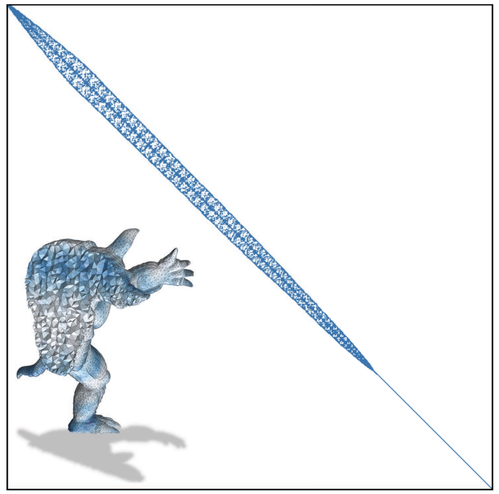
Comparison with conventional solvers

Meshless FEM solvers also do not require a volume mesh

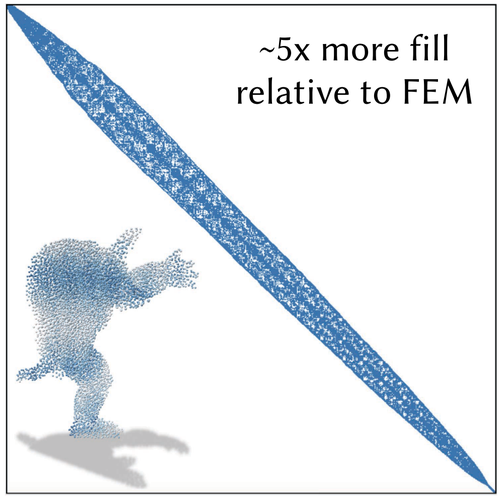
- Require dense sampling of the entire domain



- Require solving large linear systems



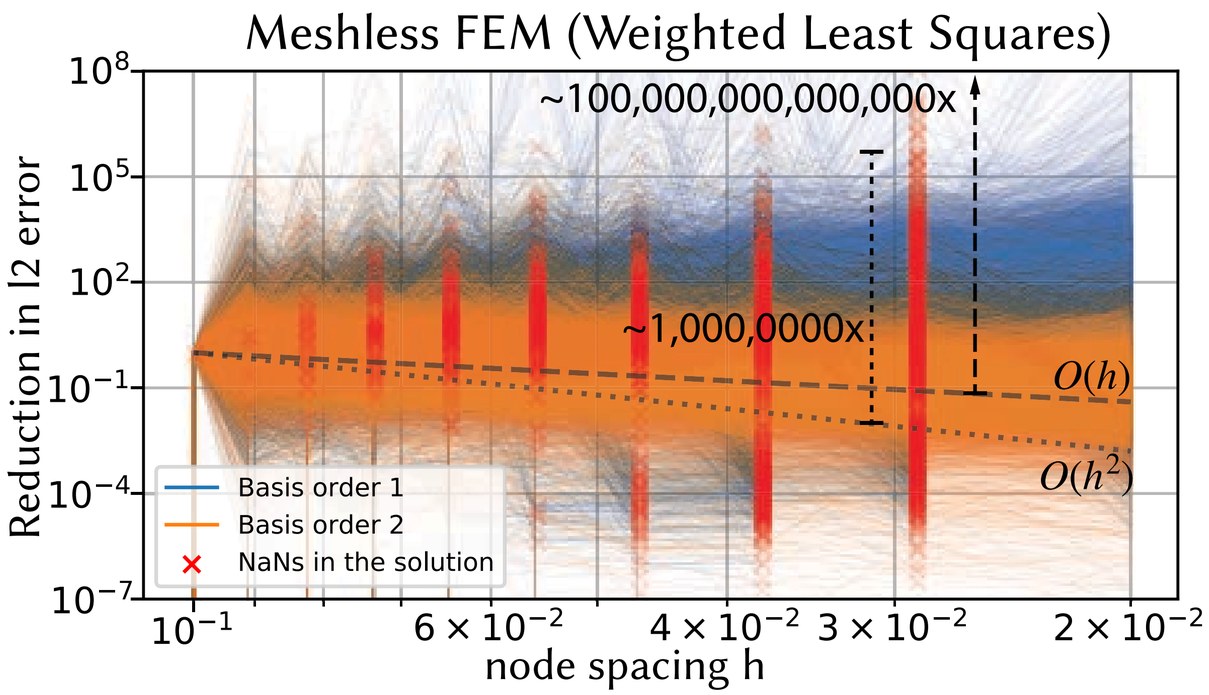
Typical FEM sparse matrix
(~25k vertices after reordering)

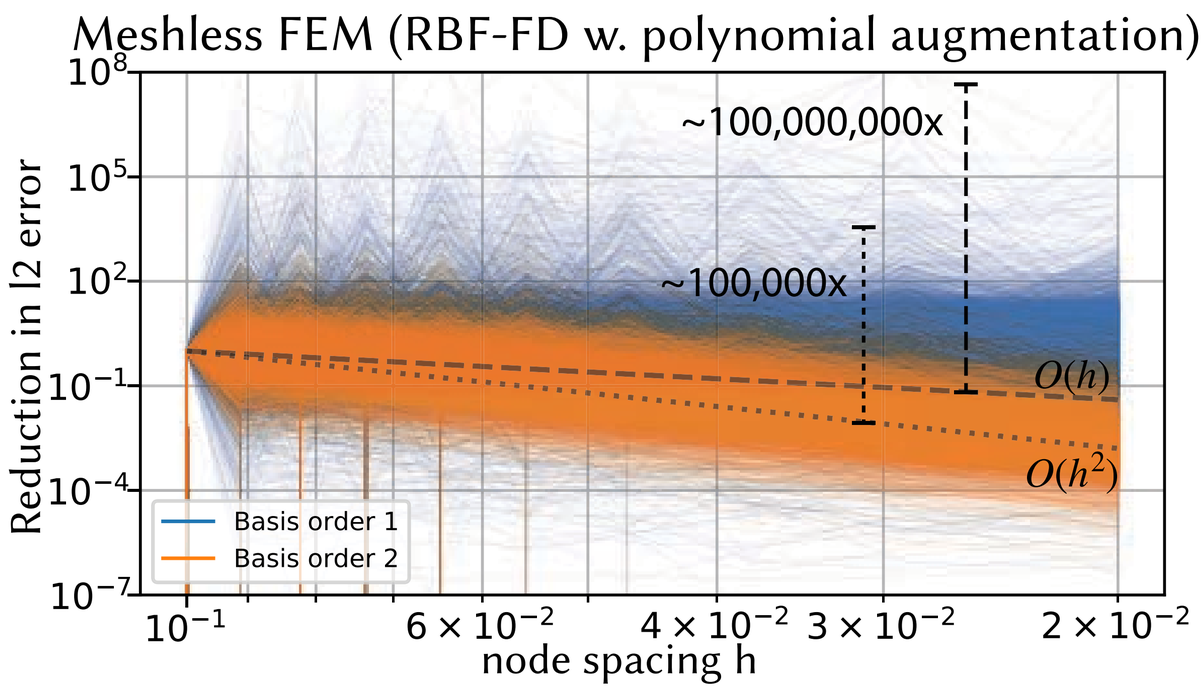


Typical Meshless FEM sparse matrix
(~25k nodes after reordering)

Meshless FEM is unreliable

Solvers have unpredictable convergence under refinement

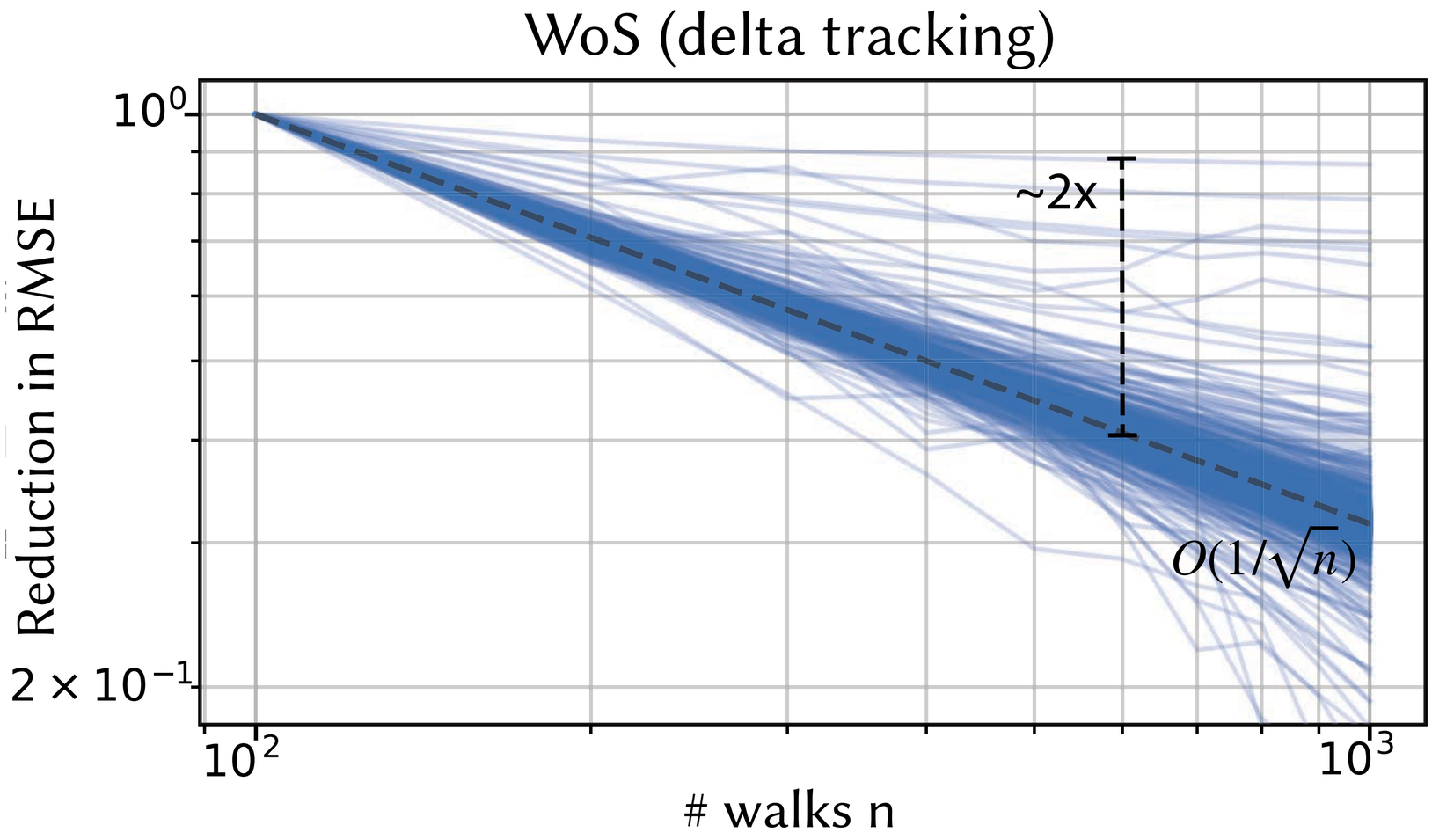




Tested on **10k models** from the Thingi10k dataset

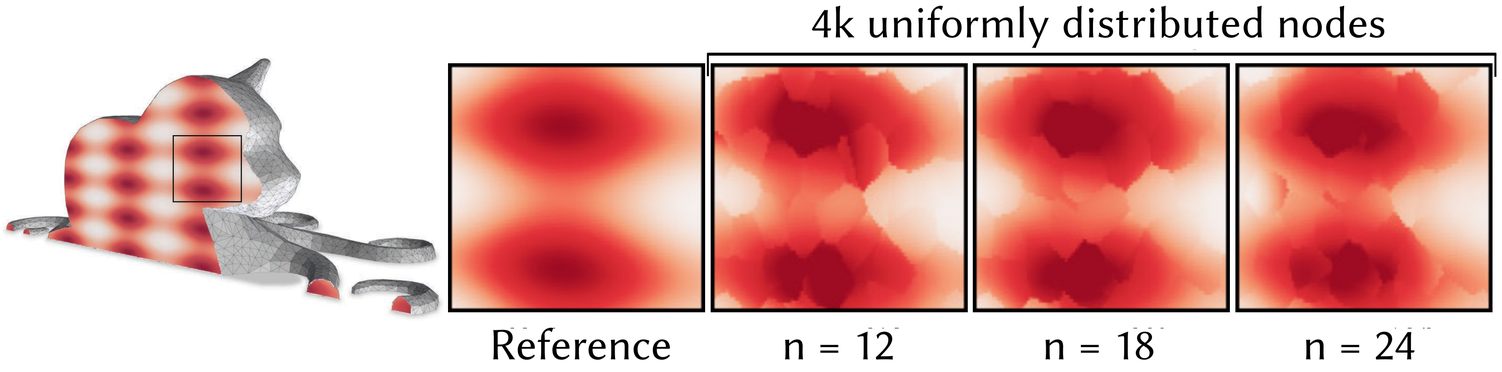
Meshless FEM is unreliable

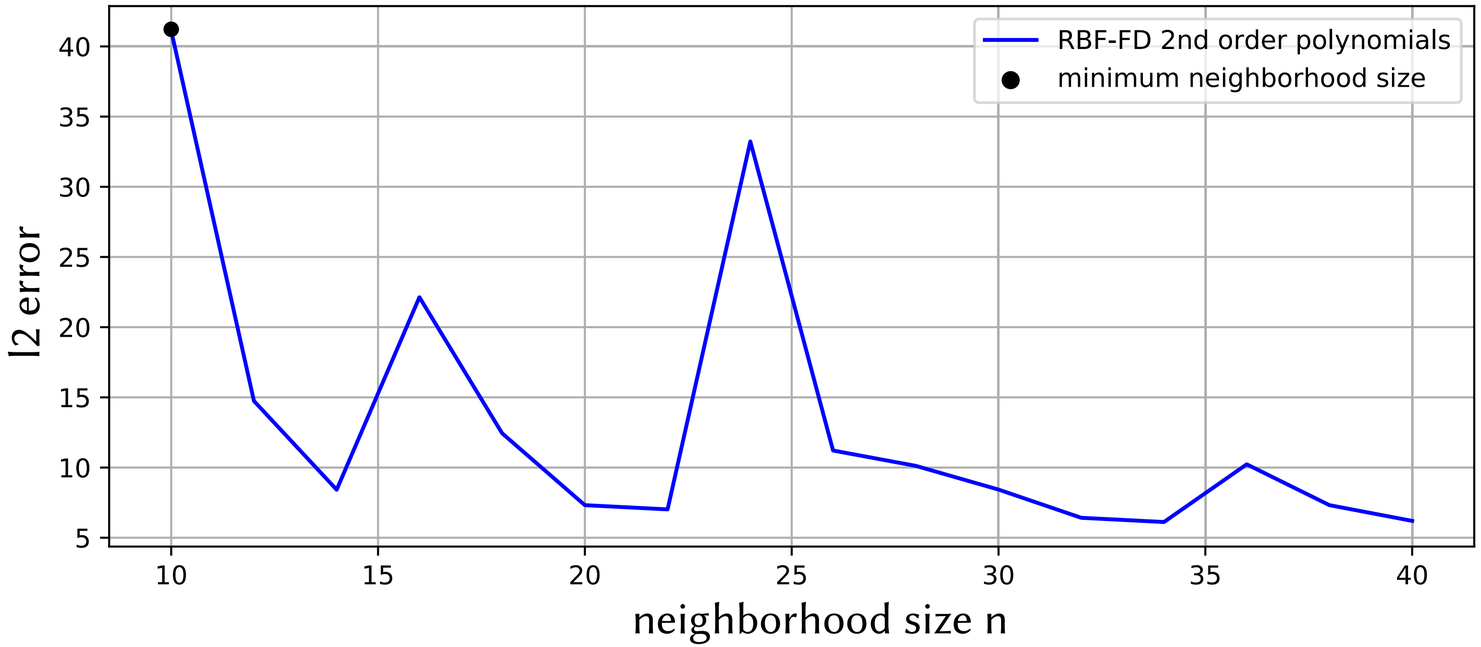
Walk on spheres converges predictably



Meshless FEM is unreliable

Solvers are difficult to tune

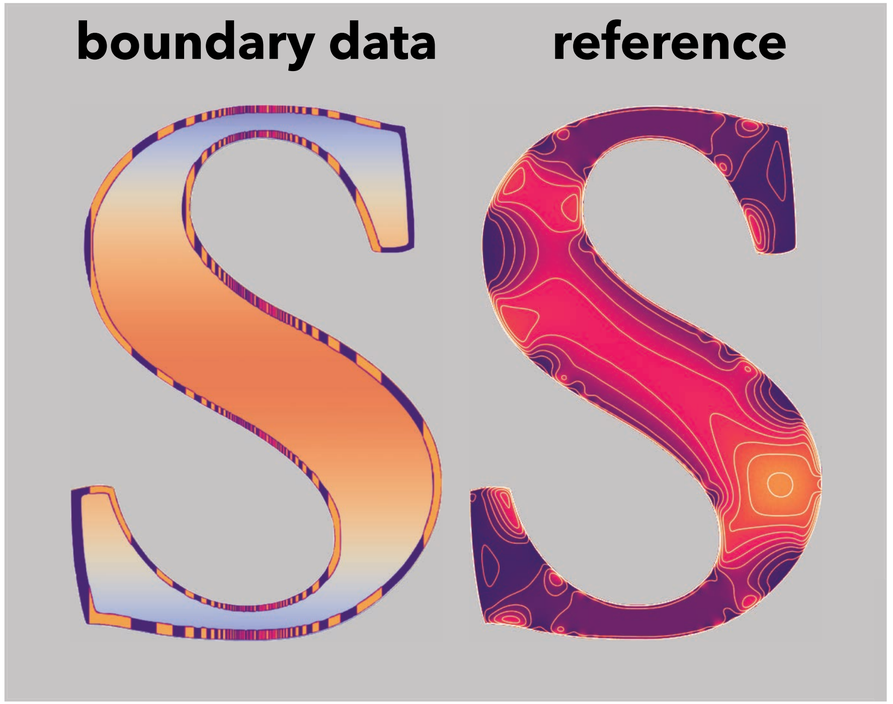


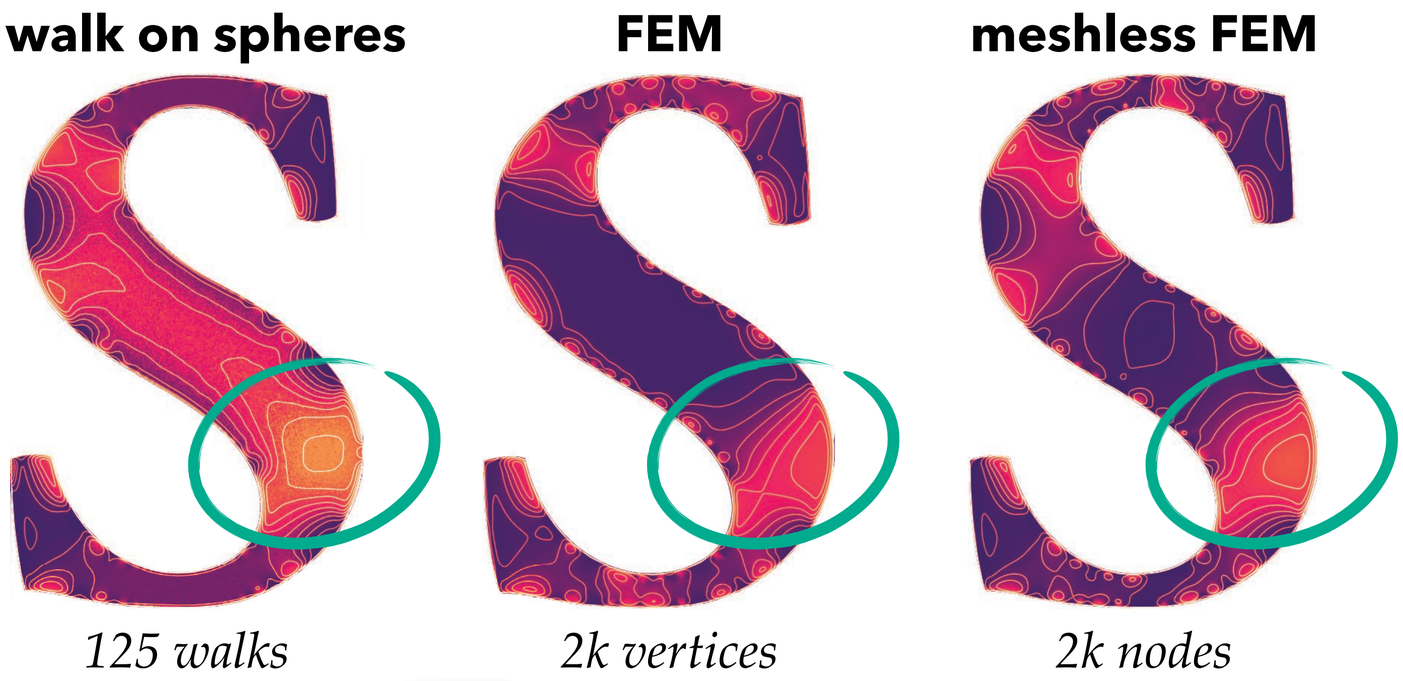


Aliasing of boundary conditions

Monte Carlo decouples boundary conditions/coefficients from geometry

Experiment: solve screened Poisson equation w/ high-frequency boundary data



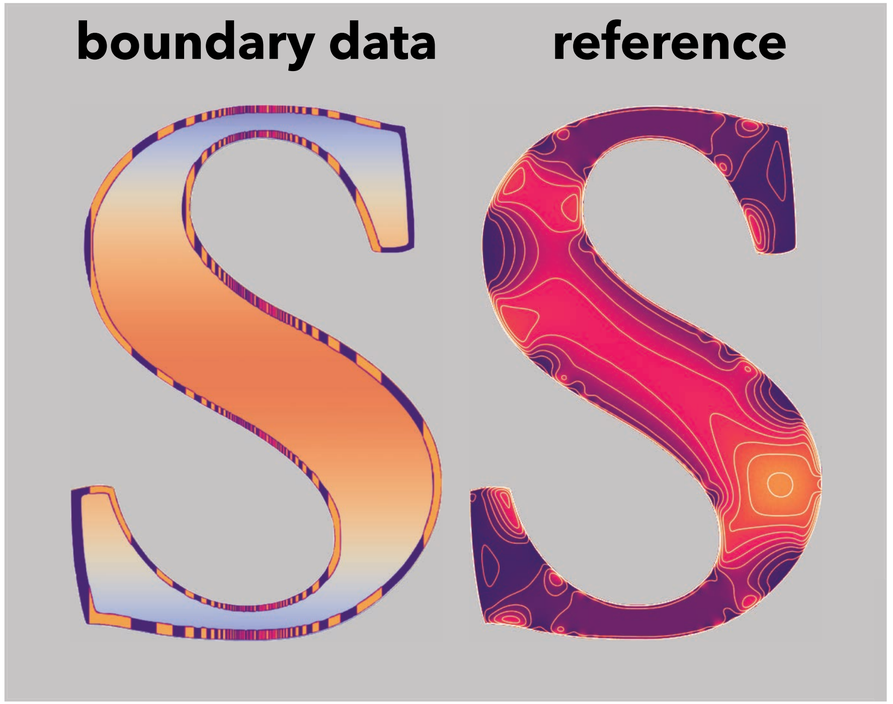


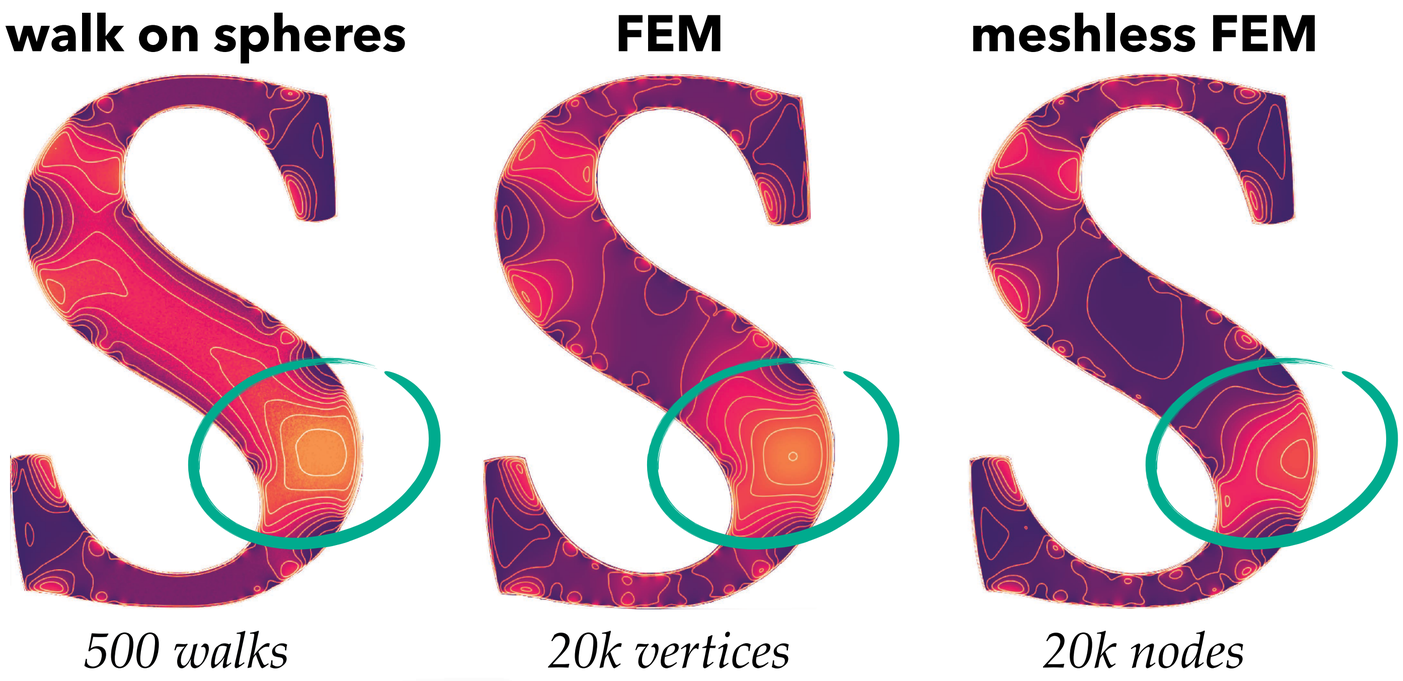
$\mathbb{E}[I_N] = I$
exact solution on average—even for N=1

Aliasing of boundary conditions

Monte Carlo decouples boundary conditions/coefficients from geometry

Experiment: solve screened Poisson equation w/ high-frequency boundary data



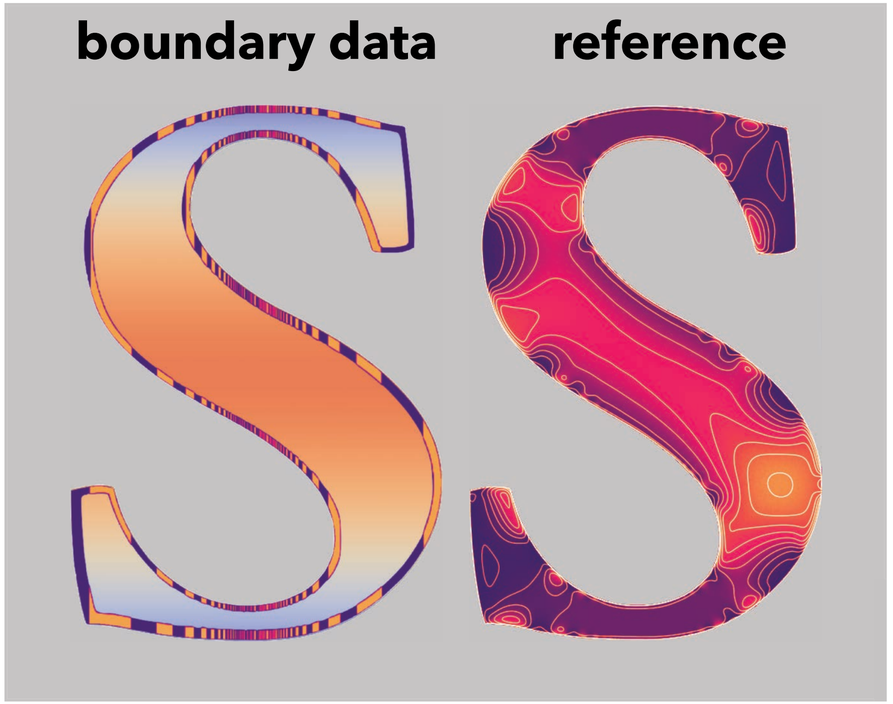


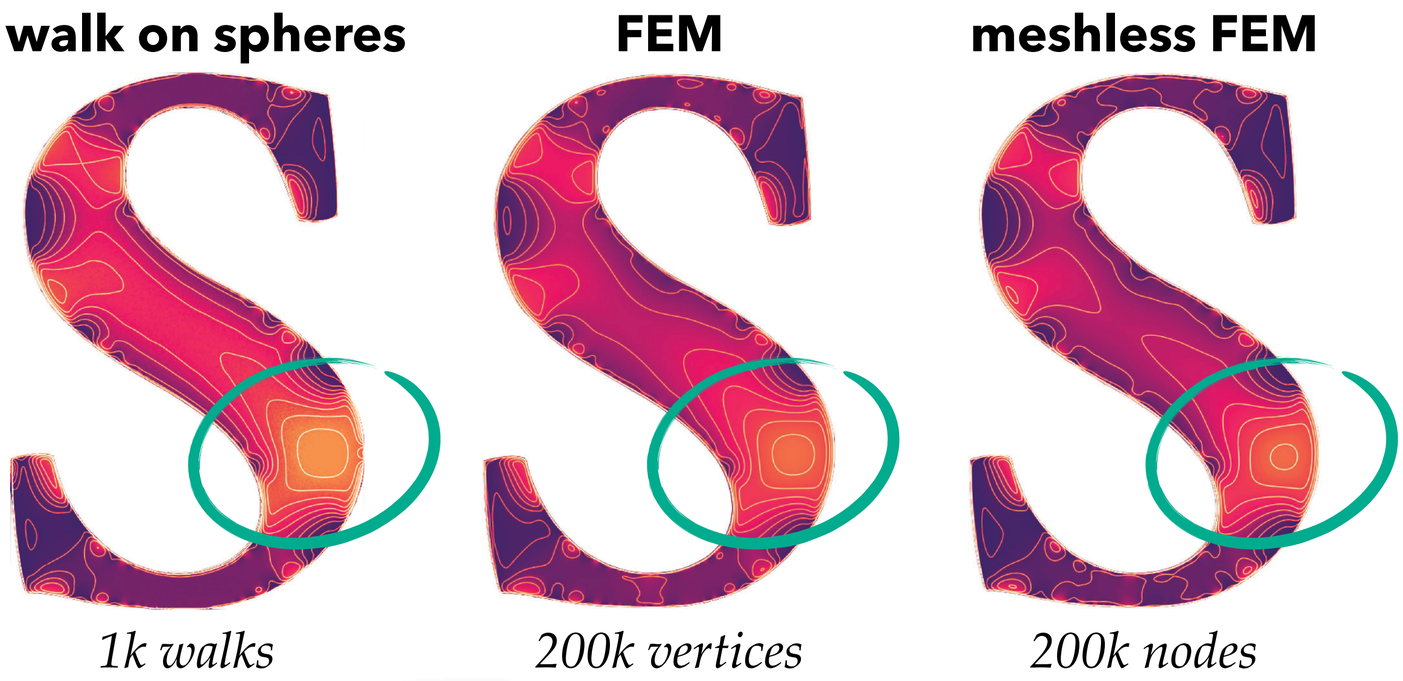
$\mathbb{E}[I_N] = I$
exact solution on average—even for N=1

Aliasing of boundary conditions

Monte Carlo decouples boundary conditions/coefficients from geometry

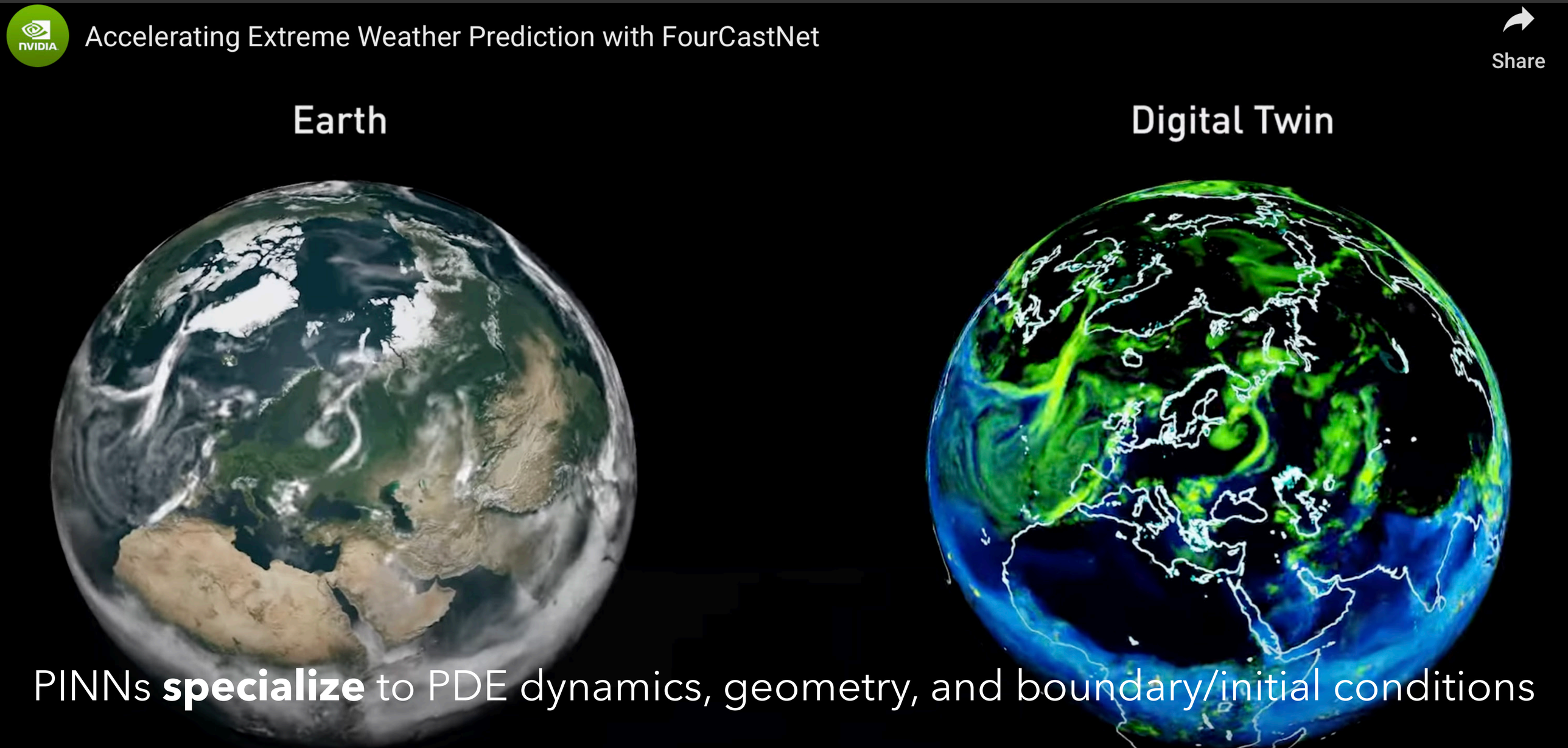
Experiment: solve screened Poisson equation w/ high-frequency boundary data





$\mathbb{E}[I_N] = I$
exact solution on average—even for N=1

Physically Informed Neural Networks (PINNS)



PINNs **specialize** to PDE dynamics, geometry, and boundary/initial conditions

AI based denoising for rendering