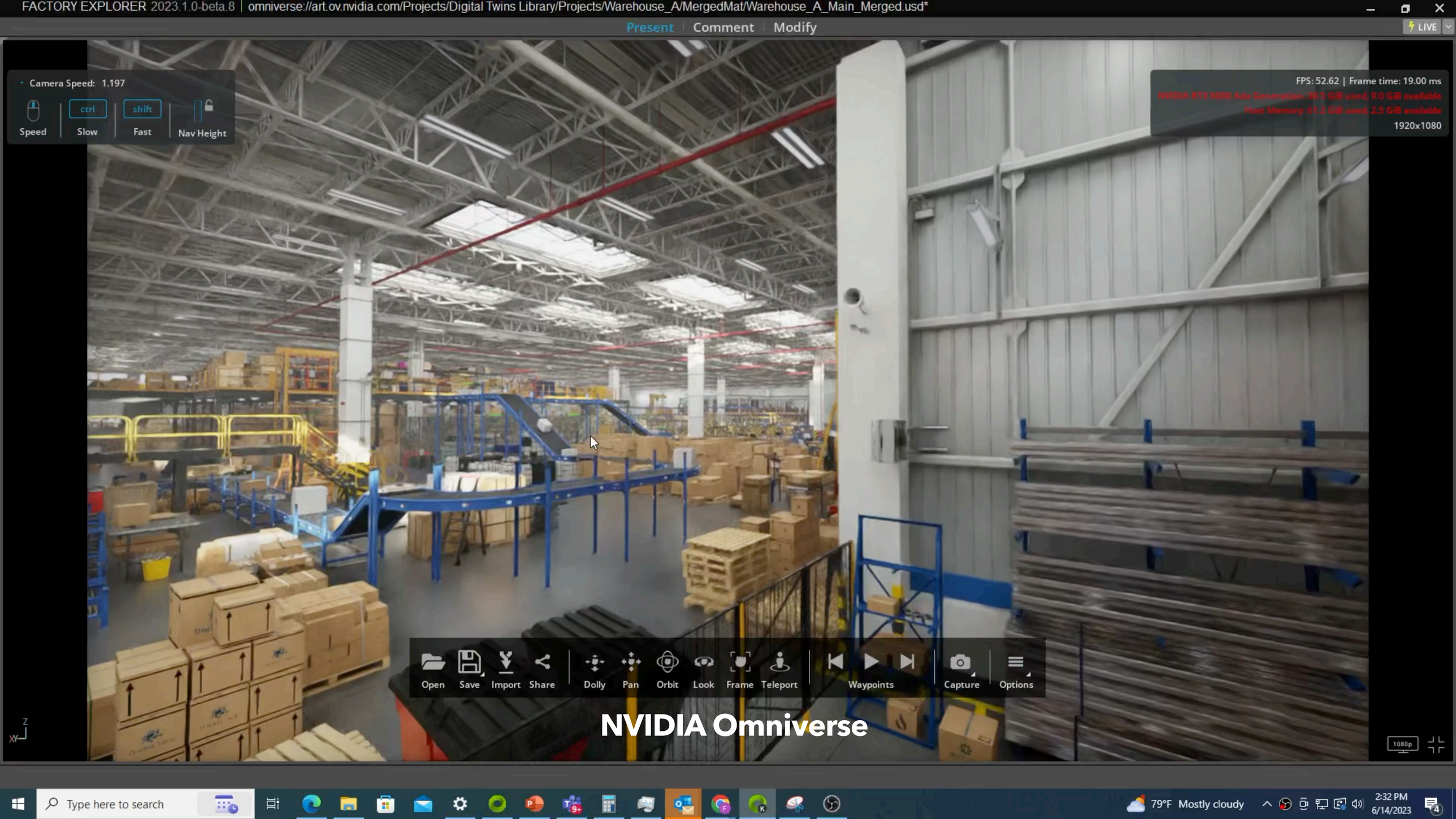
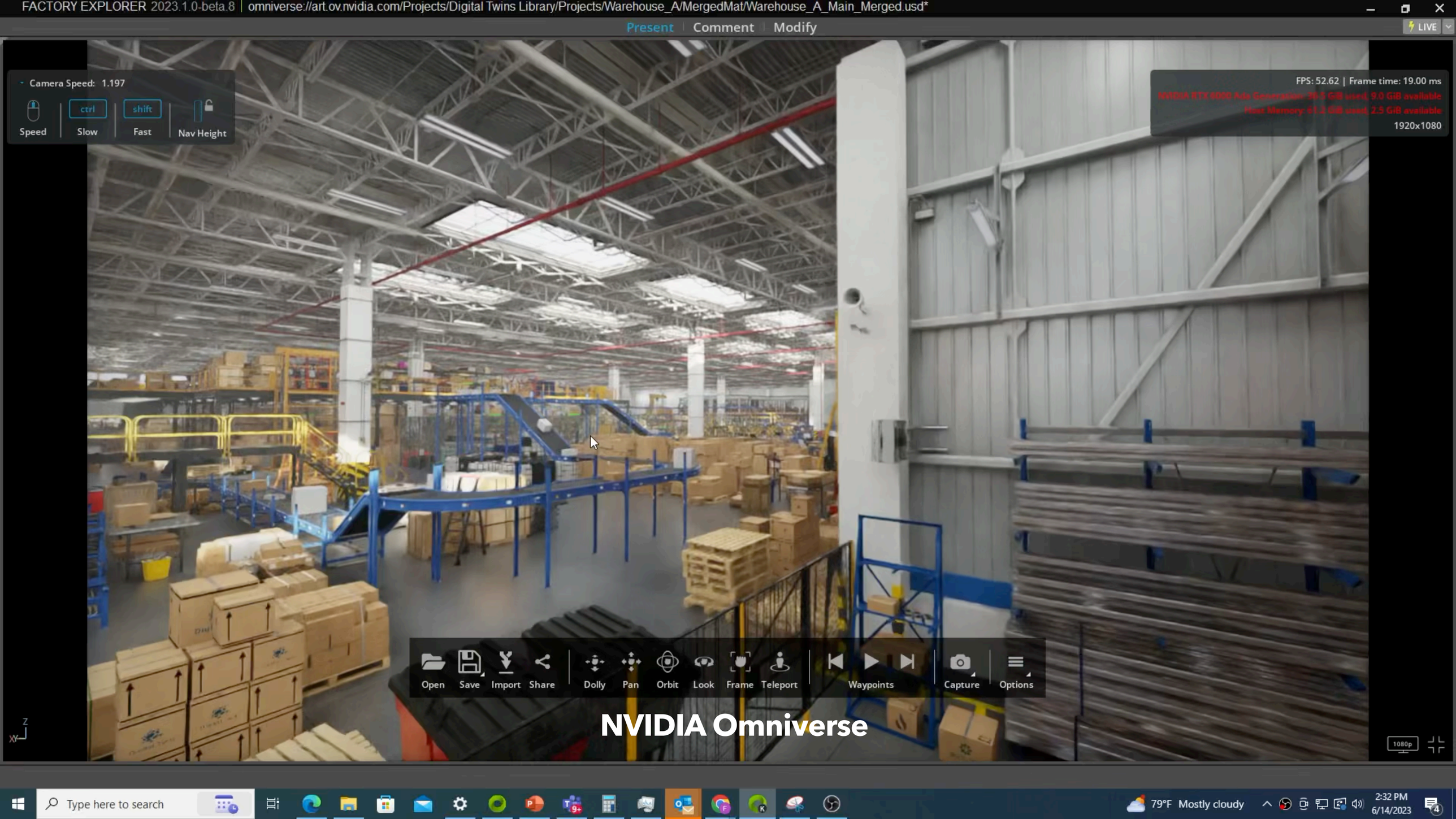
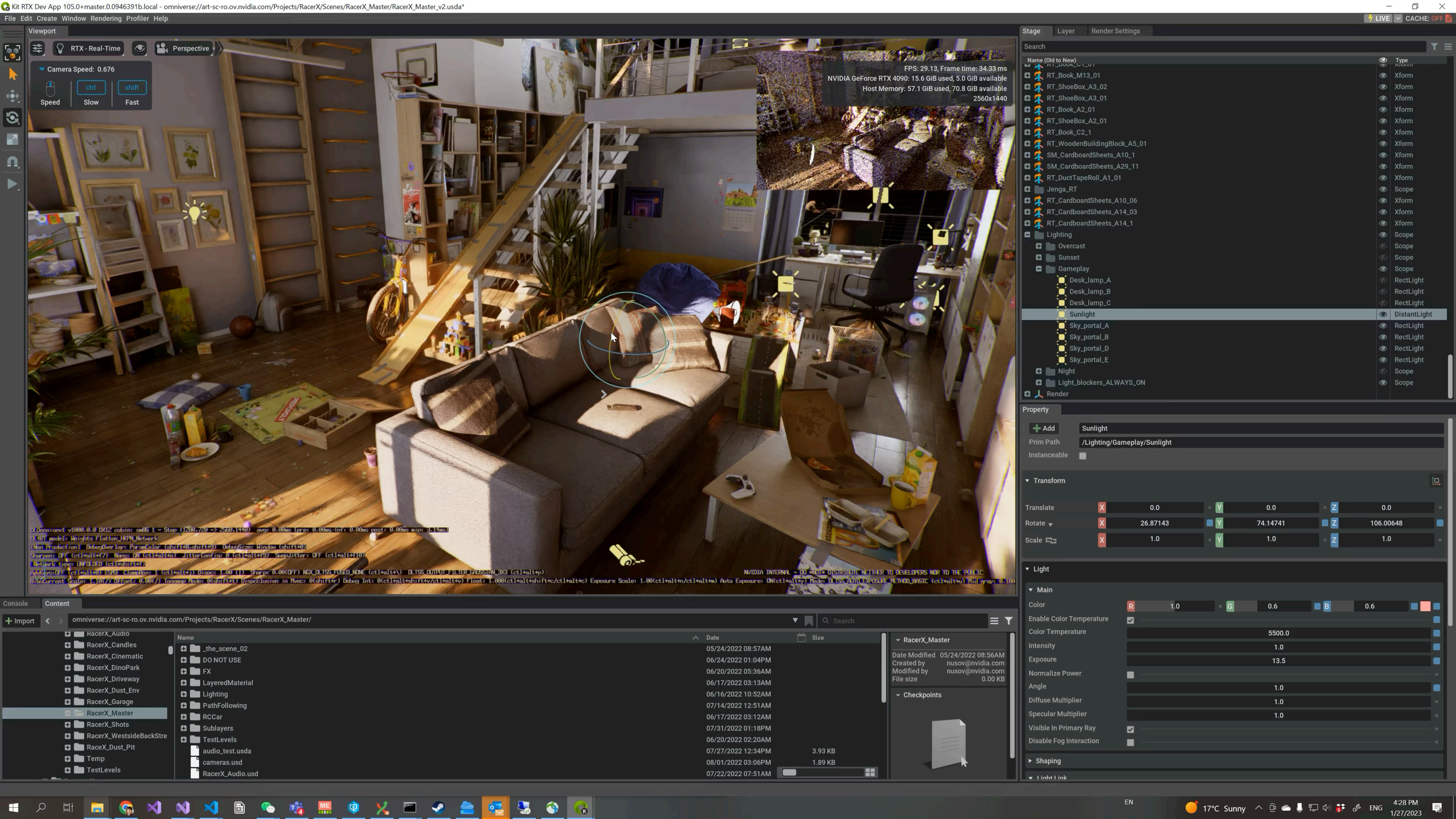
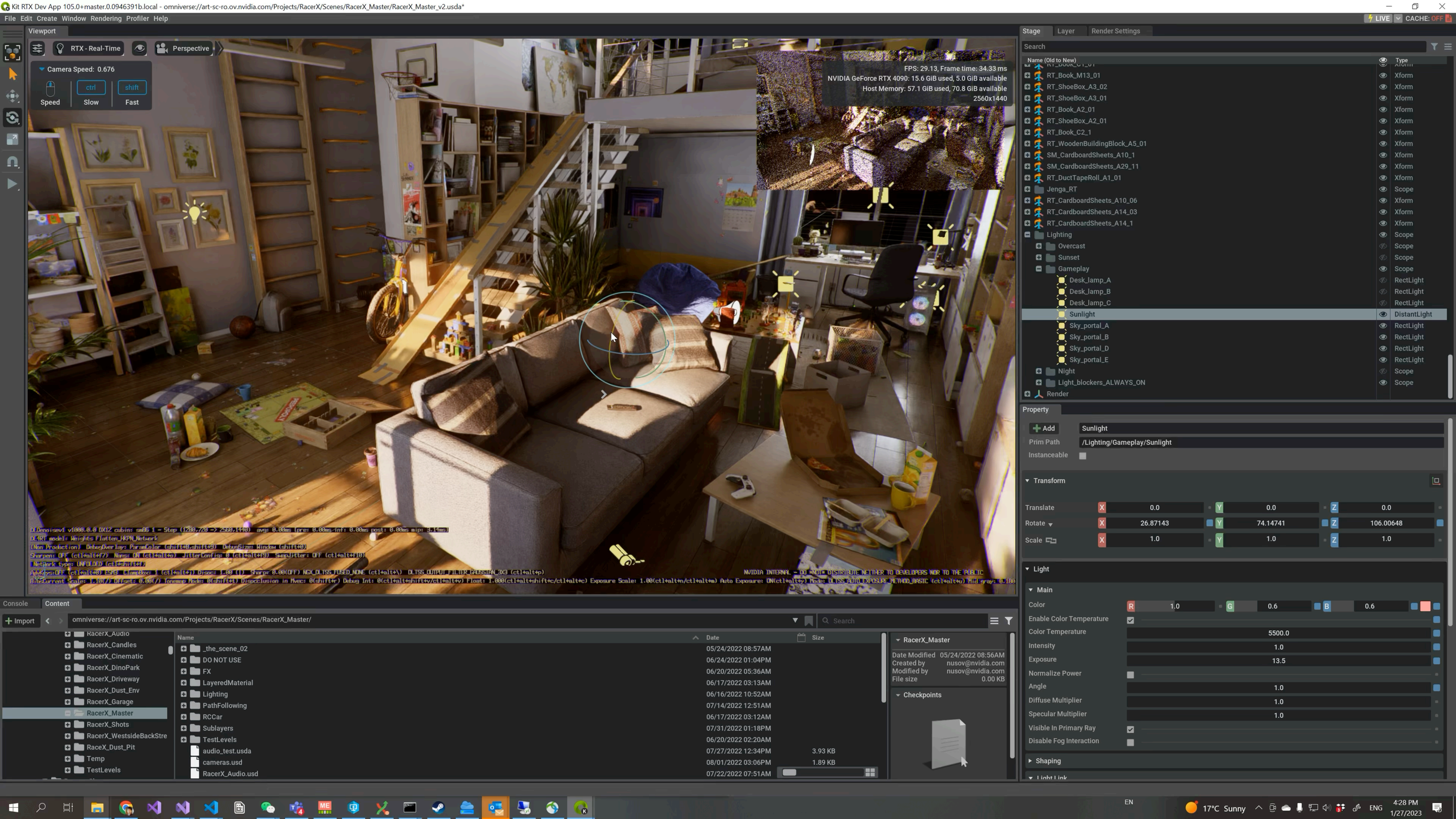
Monte Carlo Geometry Processing


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


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
Stage
Layer
Render Settings
Search
Name (Old to New)
Type
RT_Book_M13_01
RT_ShoeBox_A3_02
RT_ShoeBox_A3_01
RT_Book_A2_01
RT_ShoeBox_A2_01
RT_Book_C2_1
RT_WoodenBuildingBlock_A5_01
SM_CardboardSheets_A10_1
SM_CardboardSheets_A29_11
RT_DuctTapeRoll_A1_01
Jenga_RT
RT_CardboardSheets_A10_06
RT_CardboardSheets_A14_03
RT_CardboardSheets_A14_1
Lighting
Overcast
Sunset
Gameplay
Desk_lamp_A
Desk_lamp_B
Desk_lamp_C
Sunlight
Sky_portal_A
Sky_portal_B
Sky_portal_D
Sky_portal_E
Night
Light_blockers_ALWAYS_ON
Render
Property
Add
Sunlight
Prim Path
/Lighting/Gameplay/Sunlight
Instanceable
Transform
Translate
Rotate
Scale
0.0
26.87143
74.14741
106.00648
1.0
Light
Main
Color
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
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
EN
17°C Sunny
ENG
4:28 PM
1/27/2023


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
Stage
Layer
Render Settings
Search
Name (Old to New)
Type
RT_Book_M13_01
RT_ShoeBox_A3_02
RT_ShoeBox_A3_01
RT_Book_A2_01
RT_ShoeBox_A2_01
RT_Book_C2_1
RT_WoodenBuildingBlock_A5_01
SM_CardboardSheets_A10_1
SM_CardboardSheets_A29_11
RT_DuctTapeRoll_A1_01
Jenga_RT
RT_CardboardSheets_A10_06
RT_CardboardSheets_A14_03
RT_CardboardSheets_A14_1
Lighting
Overcast
Sunset
Gameplay
Desk_lamp_A
Desk_lamp_B
Desk_lamp_C
Sunlight
Sky_portal_A
Sky_portal_B
Sky_portal_D
Sky_portal_E
Night
Light_blockers_ALWAYS_ON
Render
Property
Add
Sunlight
Prim Path
/Lighting/Gameplay/Sunlight
Instanceable
Transform
Translate
Rotate
Scale
0.0
26.87143
74.14741
106.00648
1.0
Light
Main
Color
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
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
EN
17°C Sunny
ENG
4:28 PM
1/27/2023

DLSS 3 with Optical Multi Frame Generation

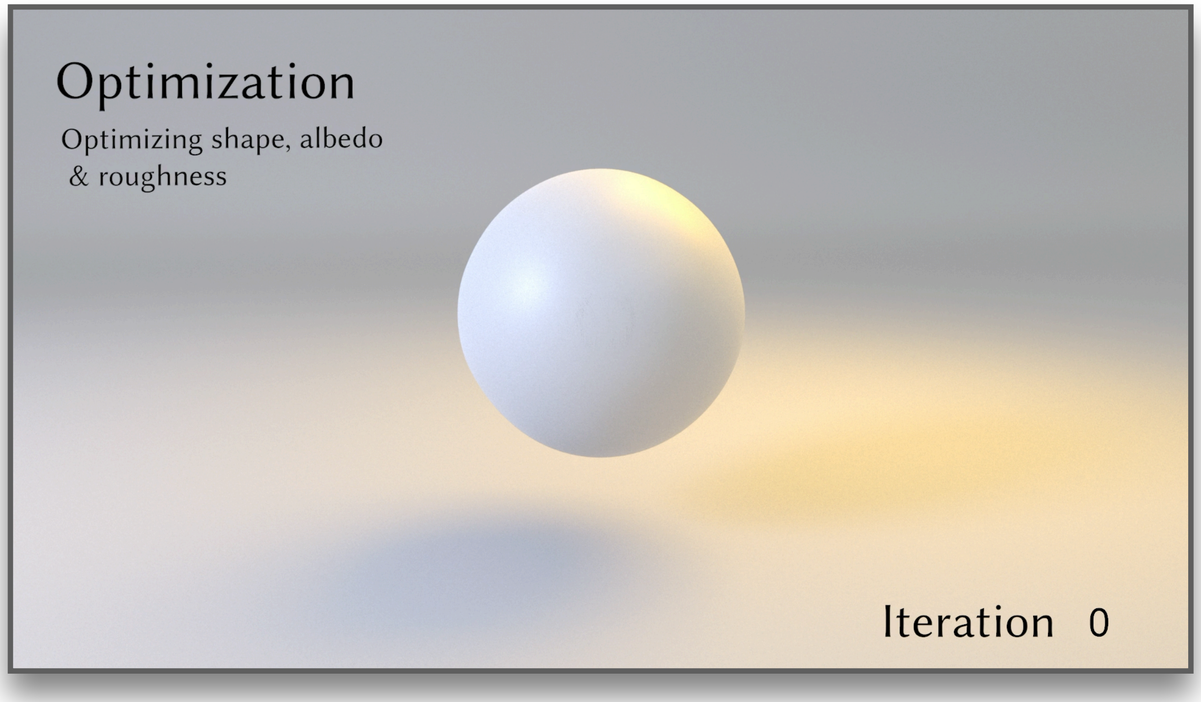
Boosts performance by using AI to generate more frames. DLSS analyzes sequential frames and motion data from the new Optical Flow Accelerator in GeForce RTX 40 Series GPUs to create additional high quality frames.

Photorealistic image generation has revolutionized industries

Entertainment

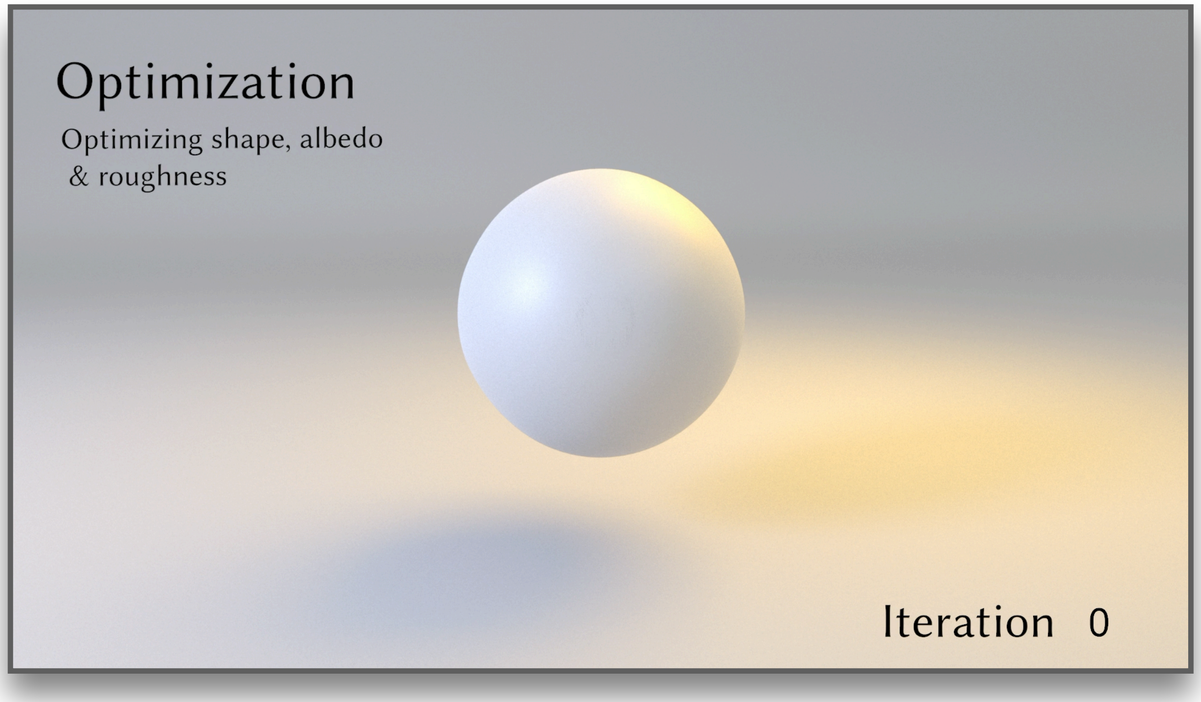
Pre-visualization

Photorealistic image generation will revolutionize more industries



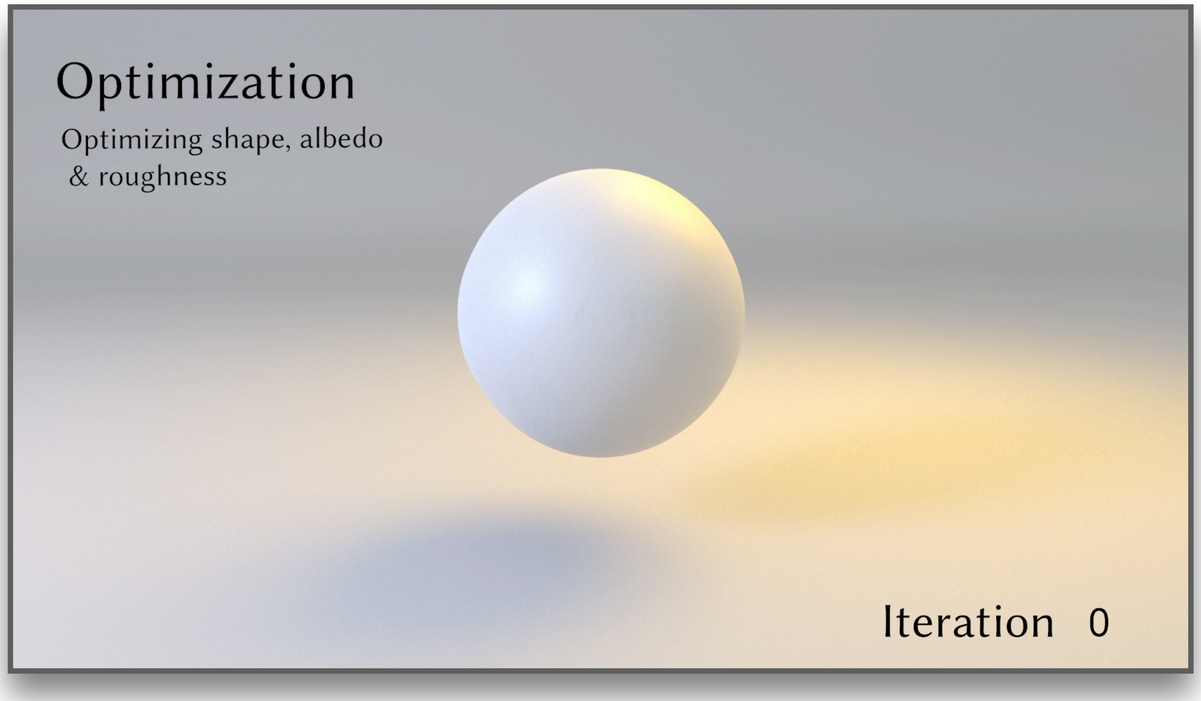
Computer Vision [Vicini et al, 2023]

Photorealistic image generation will revolutionize more industries



Computer Vision [Vicini et al, 2023]

Photorealistic image generation will revolutionize more industries

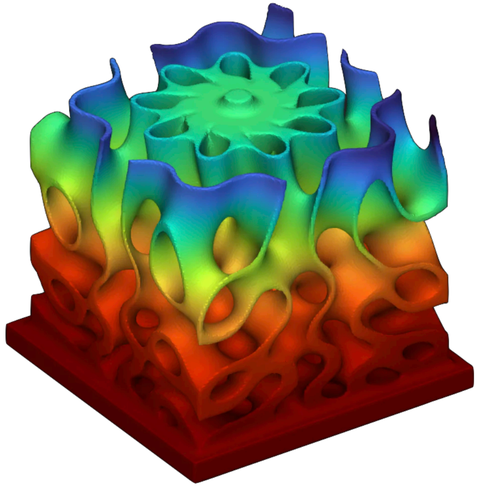


Computer Vision [Vicini et al, 2023]

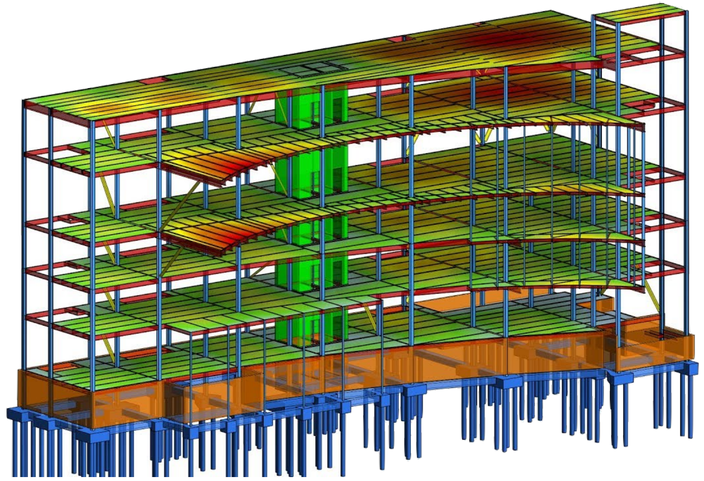
Autonomous Driving [NVIDIA Drive Sim]

Physics beyond light transport

Physics beyond light transport



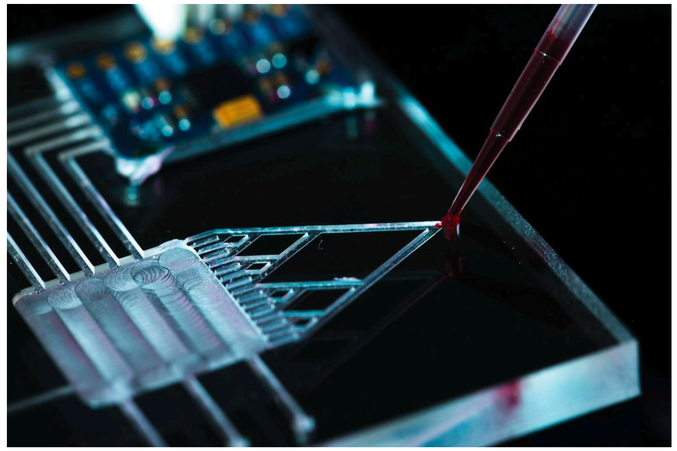
thermal diffusion



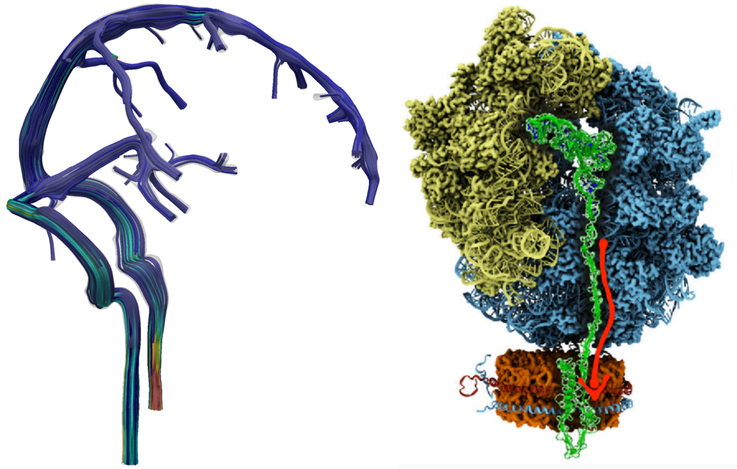
structural analysis

electrostatics

acoustic modeling

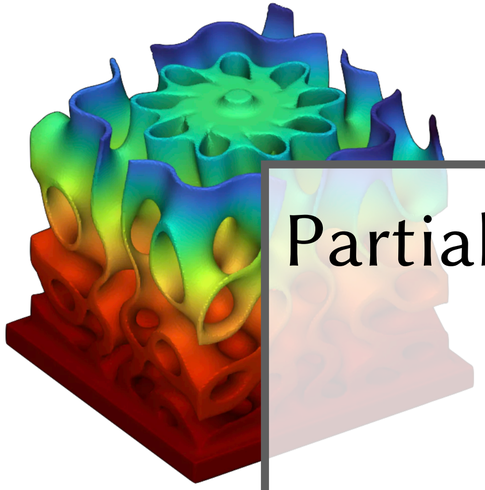


microfluidics

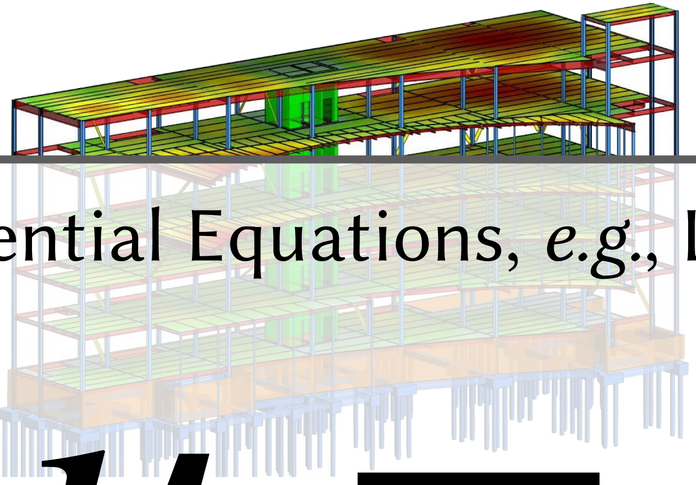


biophysics

Physics beyond light transport



thermal diffusion



structural analysis

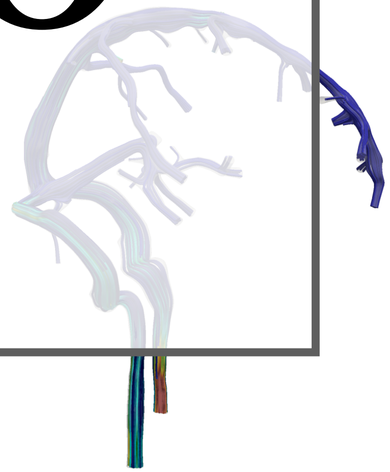
electrostatics

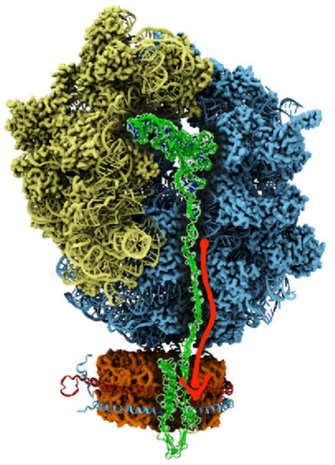
Partial Differential Equations, *e.g.*, Laplace Eq.

$$\Delta u = 0$$

acoustic modeling

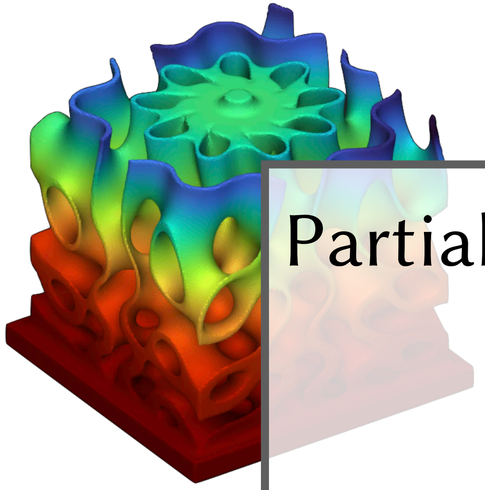
microfluidics



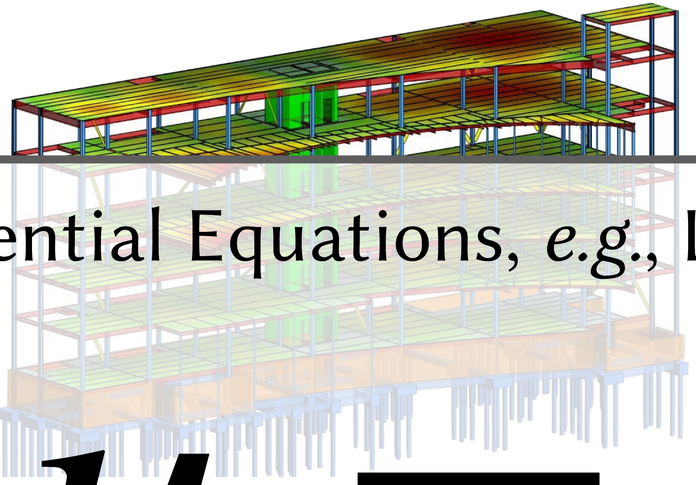


biophysics

Physics beyond light transport



thermal diffusion



structural analysis

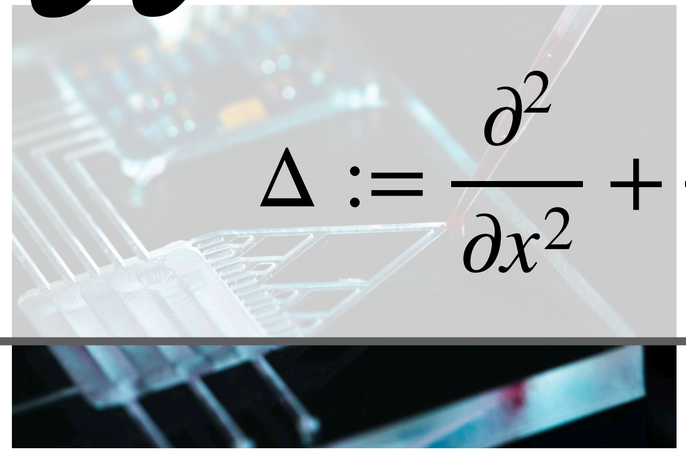
electrostatics

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

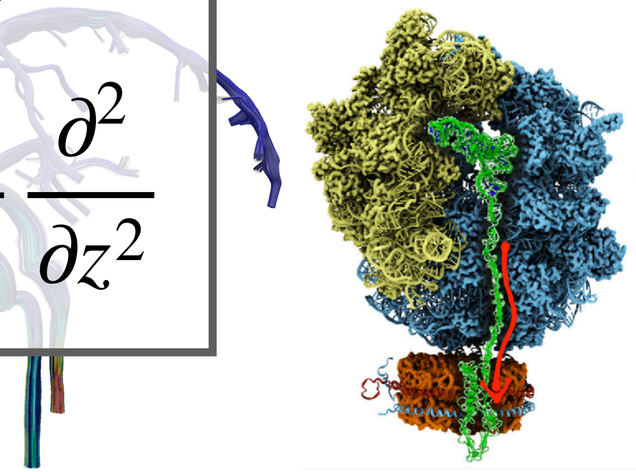
$$\Delta u = 0$$

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

acoustic modeling



microfluidics



biophysics

Partial Differential Equations (PDEs)

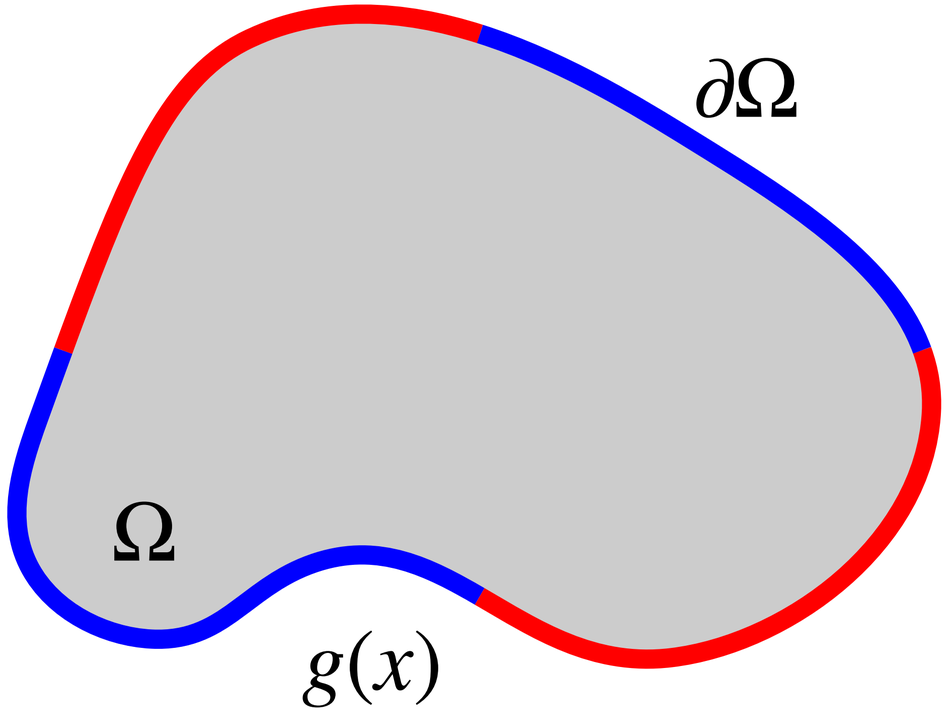
PDEs describe a function **implicitly** in terms of derivatives, solve to recover **explicit** function values

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

Partial Differential Equations (PDEs)

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

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

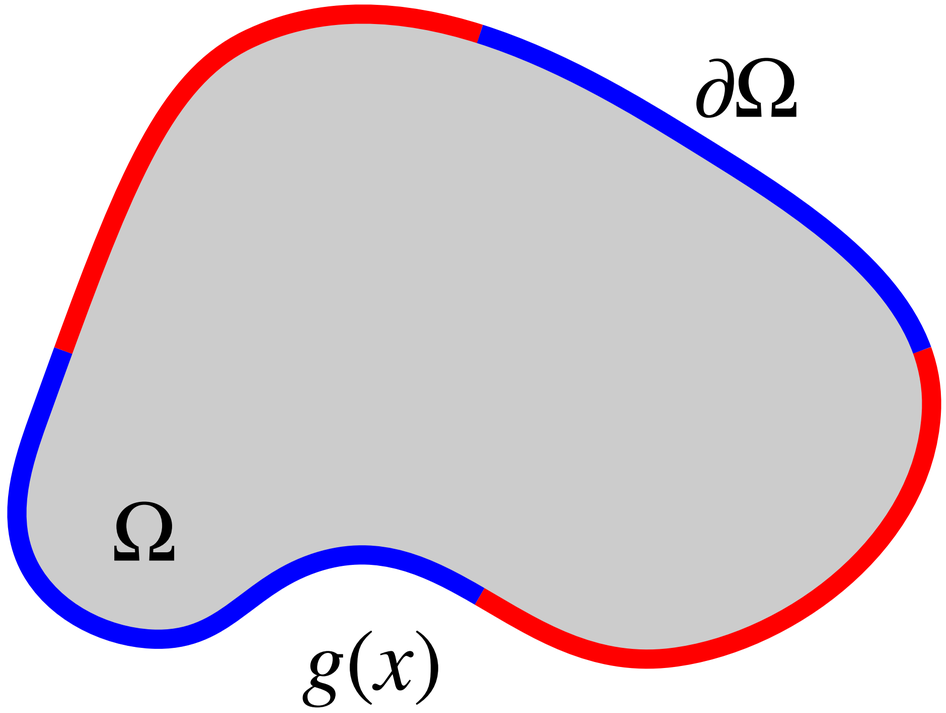


Given: values g on boundary

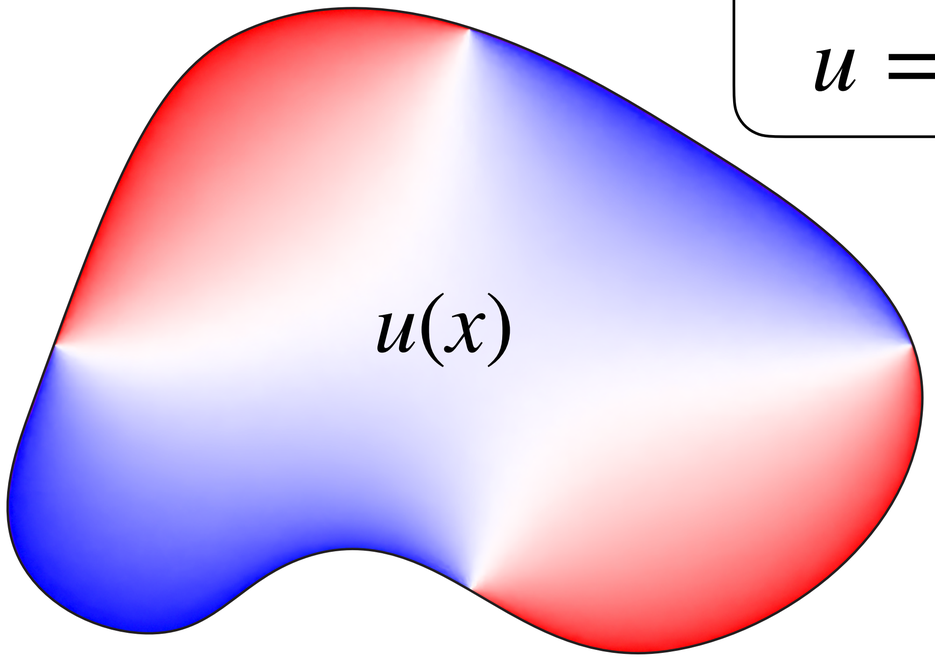
Partial Differential Equations (PDEs)

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

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

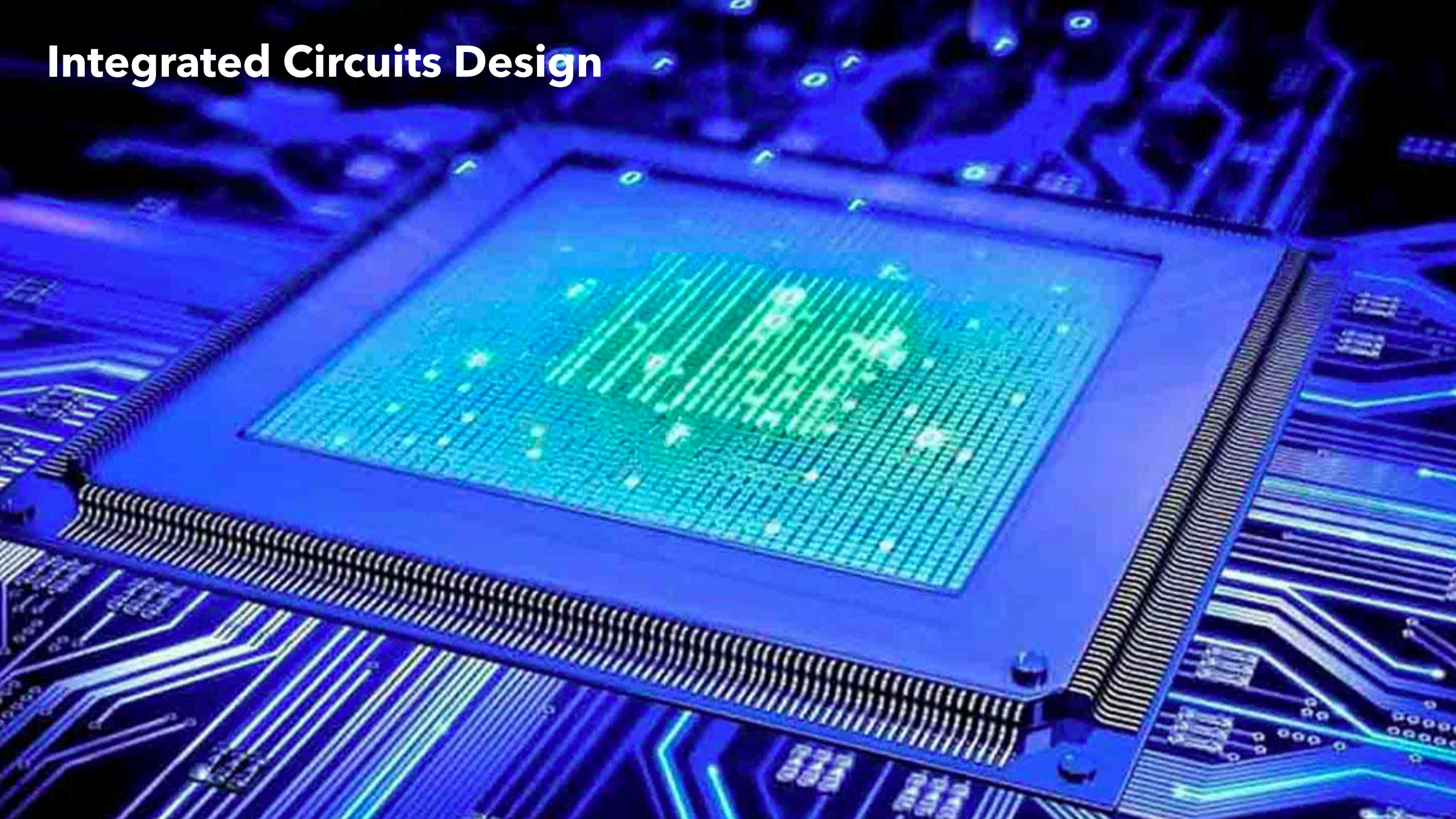


Given: values g on boundary



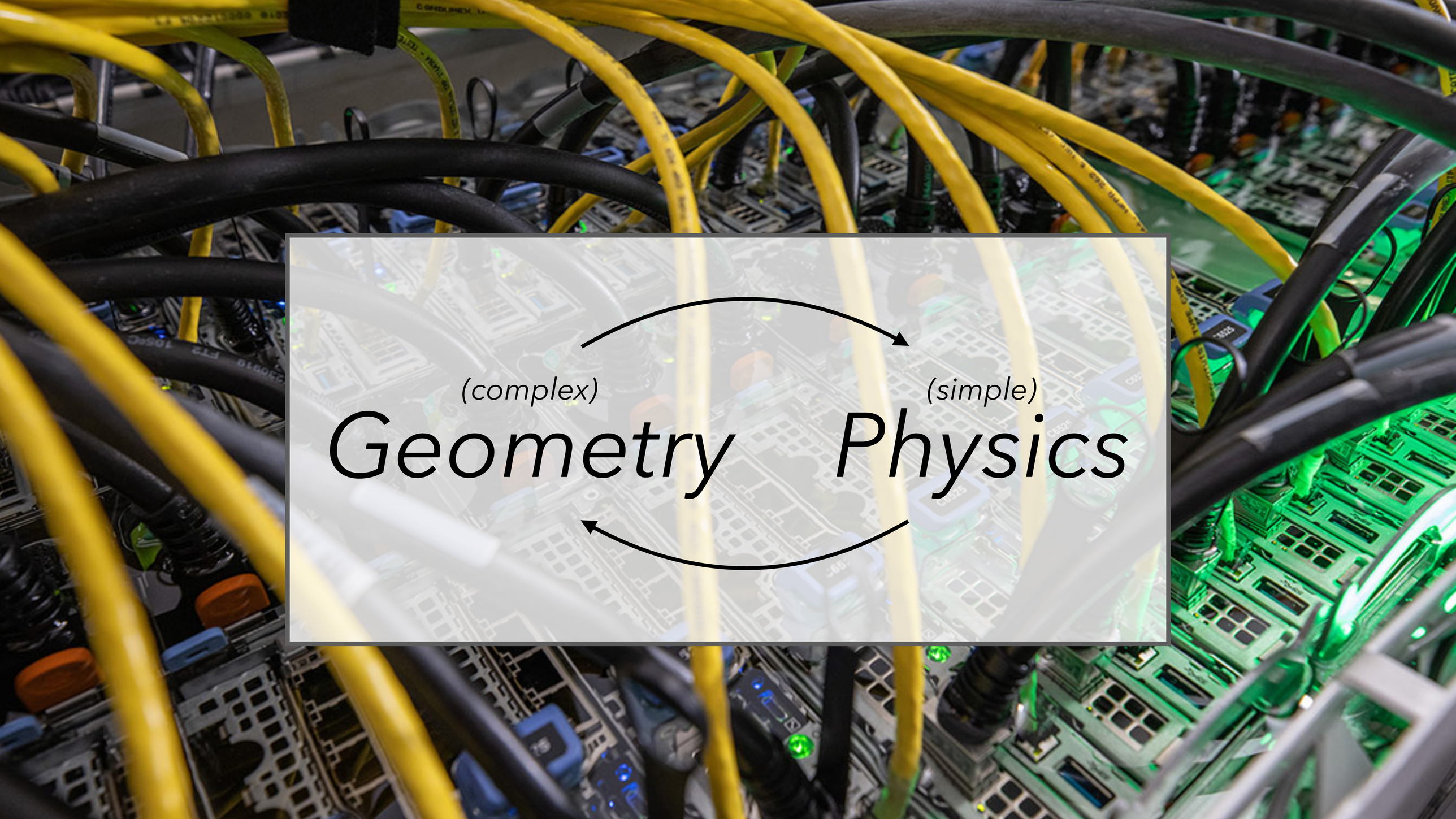
Find: solution u on interior

Integrated Circuits Design

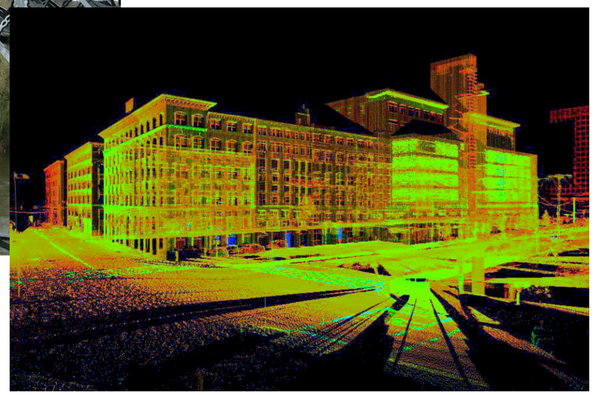


Data Center Design

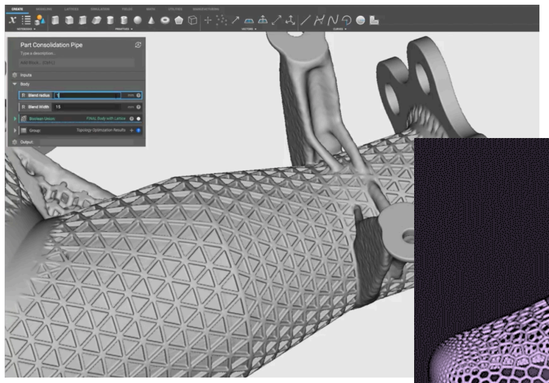
C6525

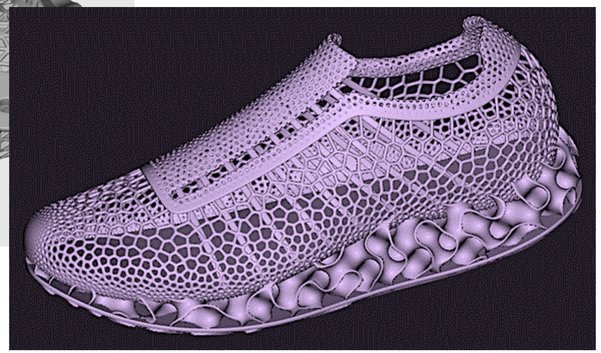

(complex)
Geometry
(simple)
Physics

Geometric complexity has increased drastically

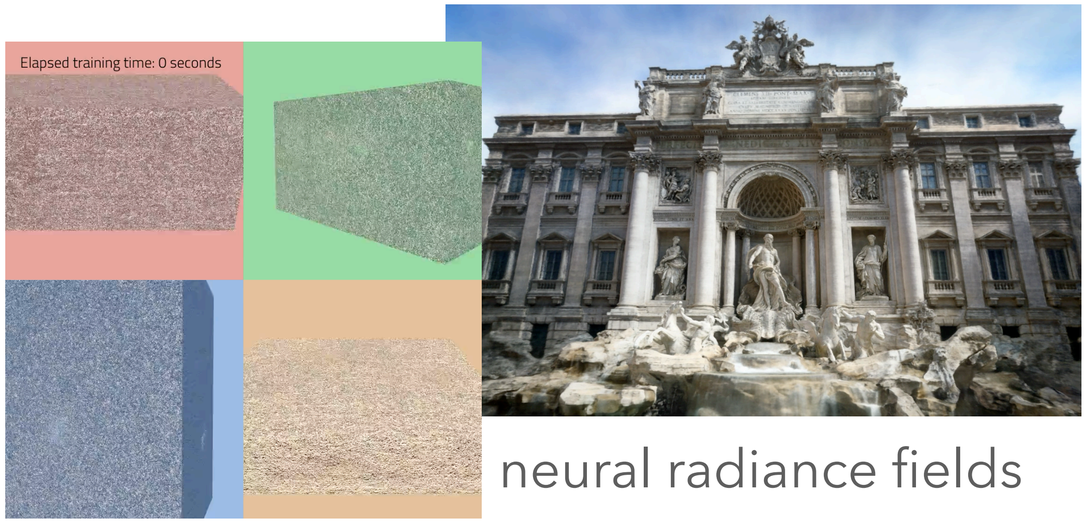


laser scanning / LiDAR

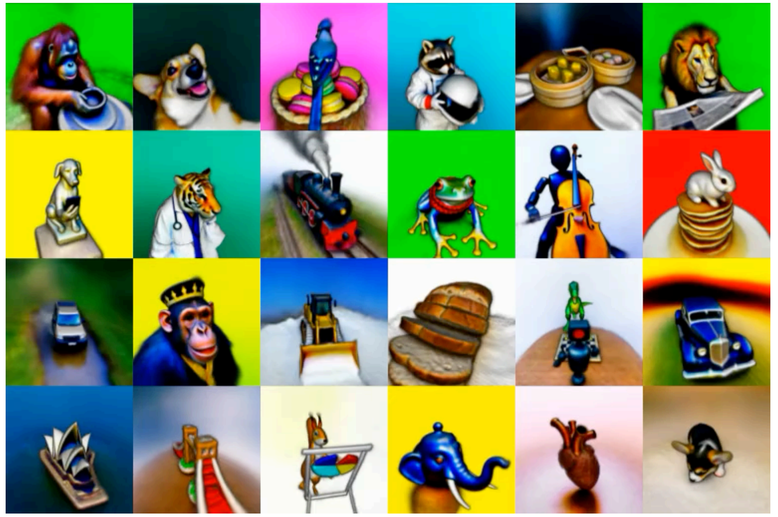




additive manufacturing

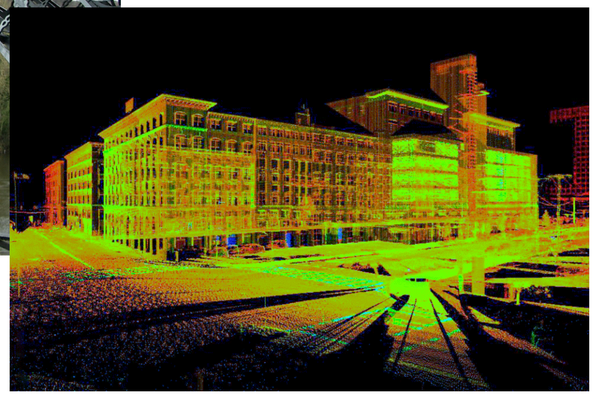


neural radiance fields

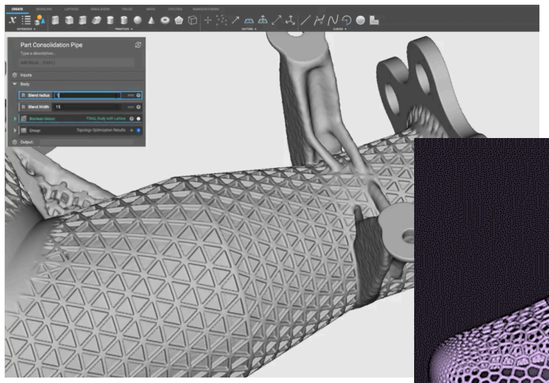


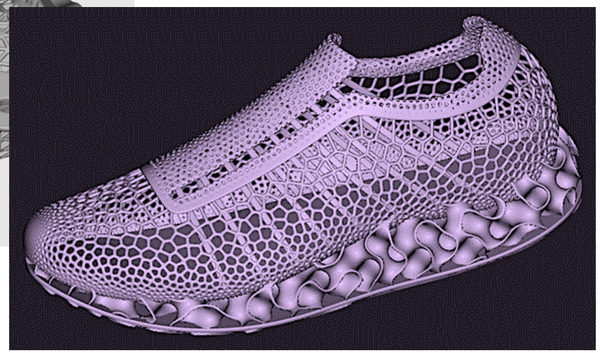
text-to-3D

Geometric complexity has increased drastically

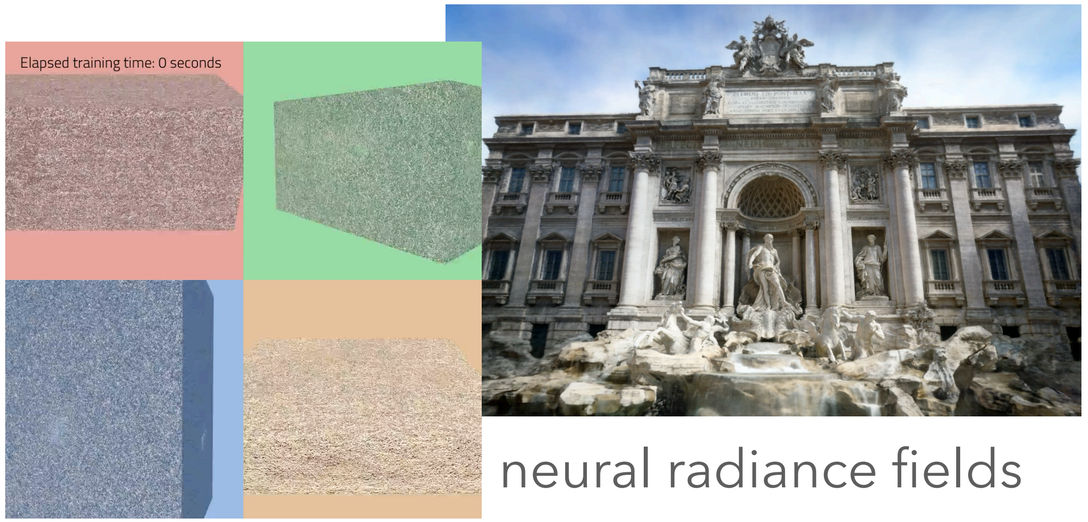


laser scanning / LiDAR

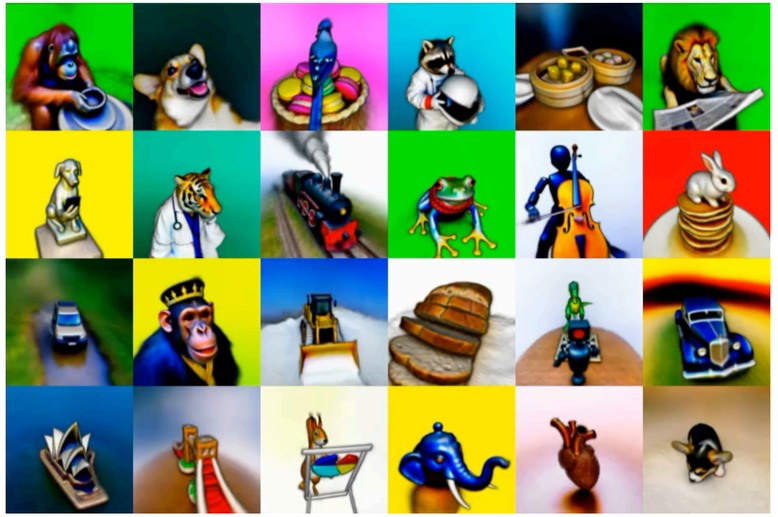




additive manufacturing



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

– …

Traditional methods for solving PDEs

“Zoo” of solvers:

– finite difference methods

– finite element methods

– finite volume methods

– boundary element methods

– spectral methods

– …

Common thread: all use *finite dimensional approximation*

Traditional methods for solving PDEs

“Zoo” of solvers:

– finite difference methods

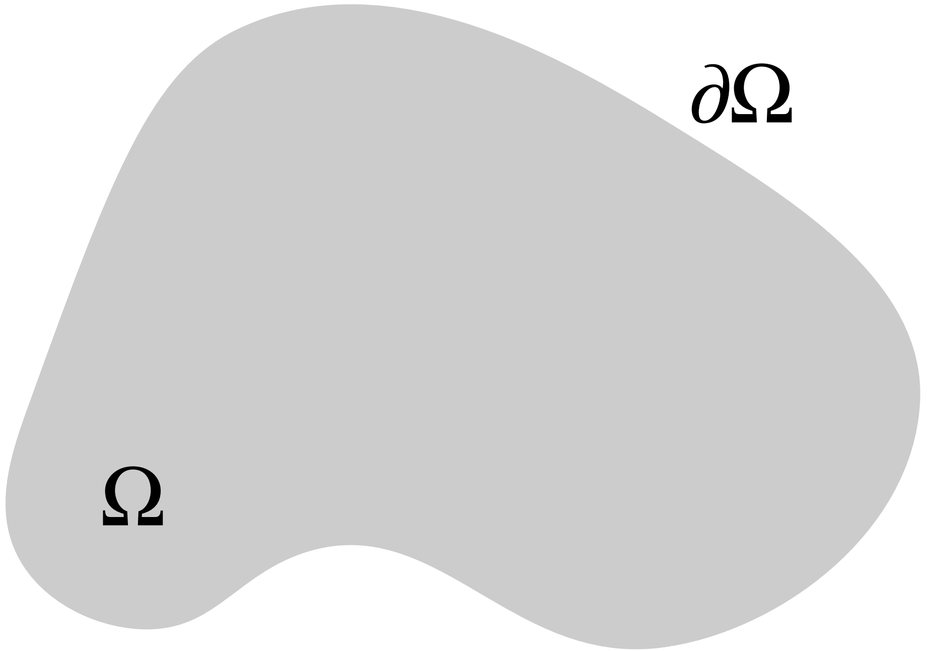
– finite element methods

– finite volume methods

– boundary element methods

– spectral methods

– …



Common thread: all use *finite dimensional approximation*

Traditional methods for solving PDEs

"Zoo" of solvers:

– finite difference methods

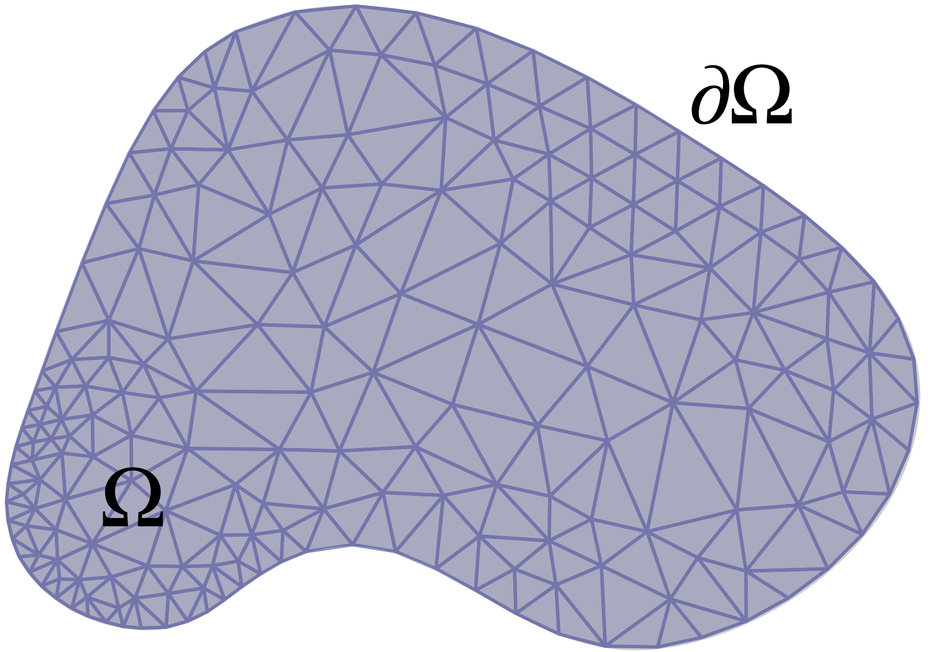
– finite element methods

– finite volume methods

– boundary element methods

– spectral methods

– …



Common thread: all use *finite dimensional approximation*

Traditional methods for solving PDEs

"Zoo" of solvers:

– finite difference methods

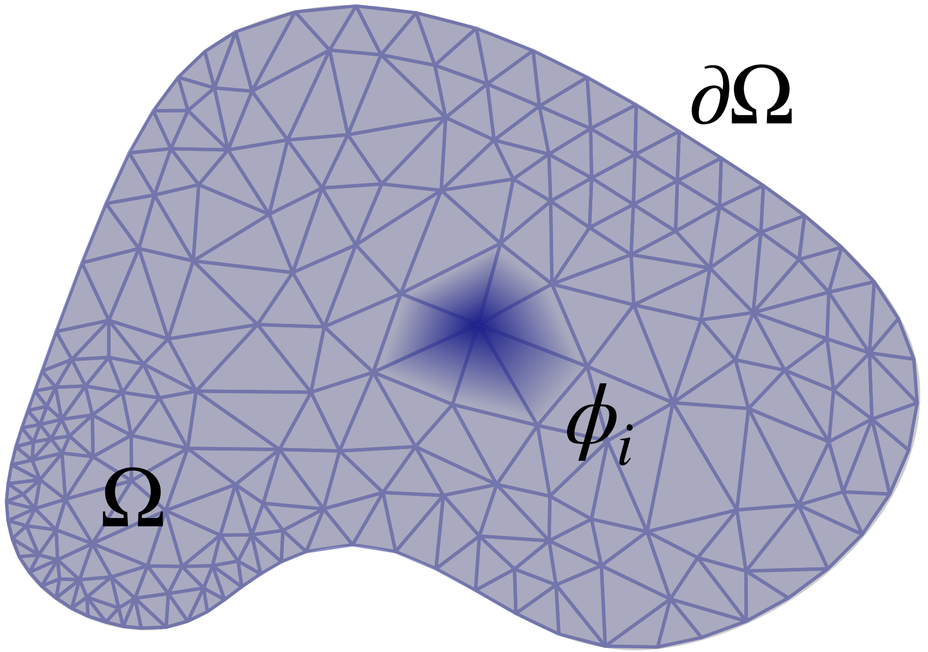
– finite element methods

– finite volume methods

– boundary element methods

– spectral methods

– …



Common thread: all use *finite dimensional approximation*

Traditional methods for solving PDEs

"Zoo" of solvers:

– finite difference methods

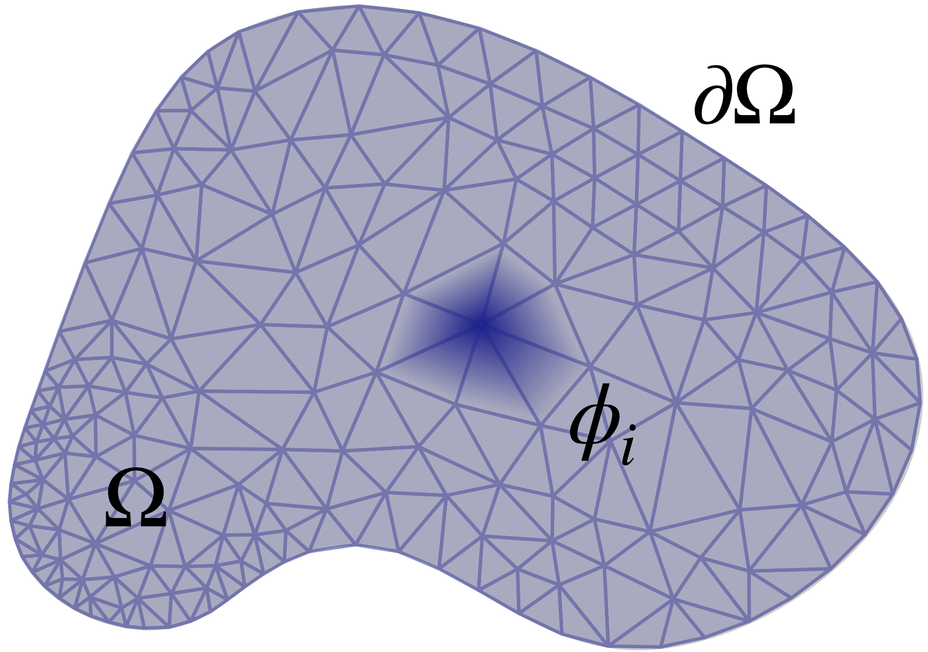
– finite element methods

– finite volume methods

– boundary element methods

– spectral methods

– …



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

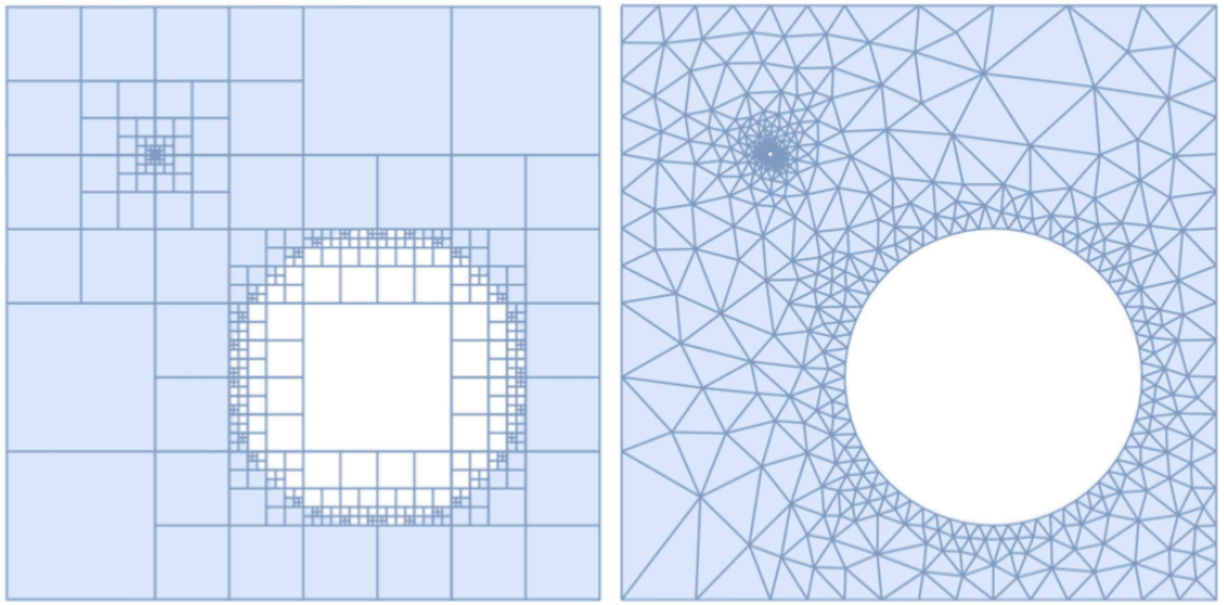
Common thread: all use *finite dimensional approximation*

Traditional methods for solving PDEs

Inevitable consequence of finite dimensional approximation:

Traditional methods for solving PDEs

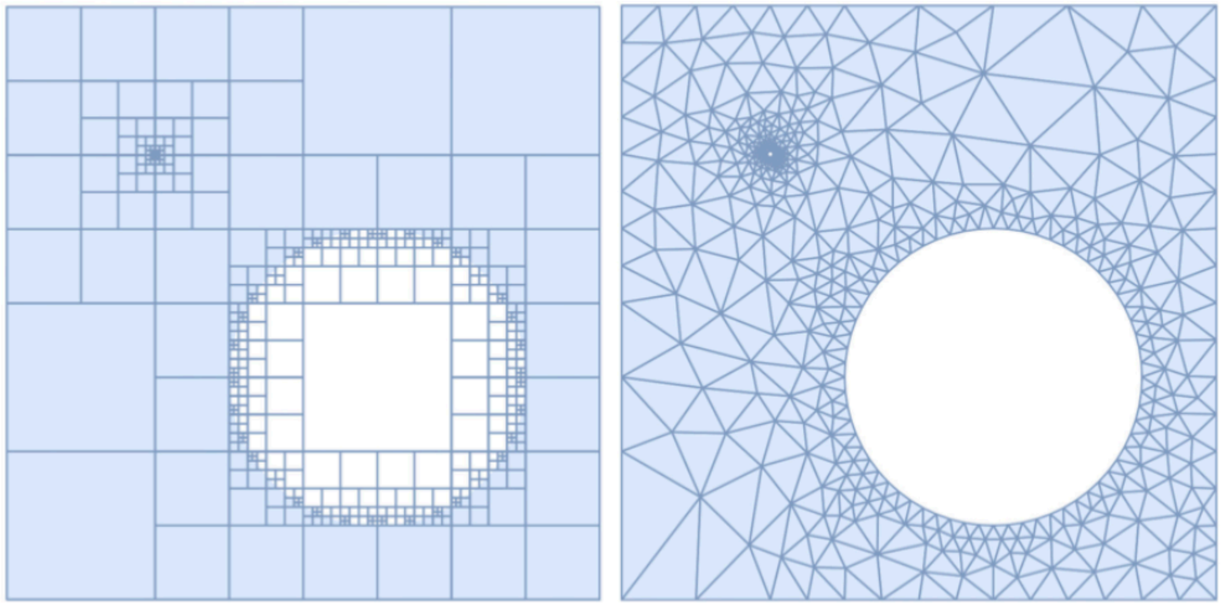
Inevitable consequence of finite dimensional approximation:



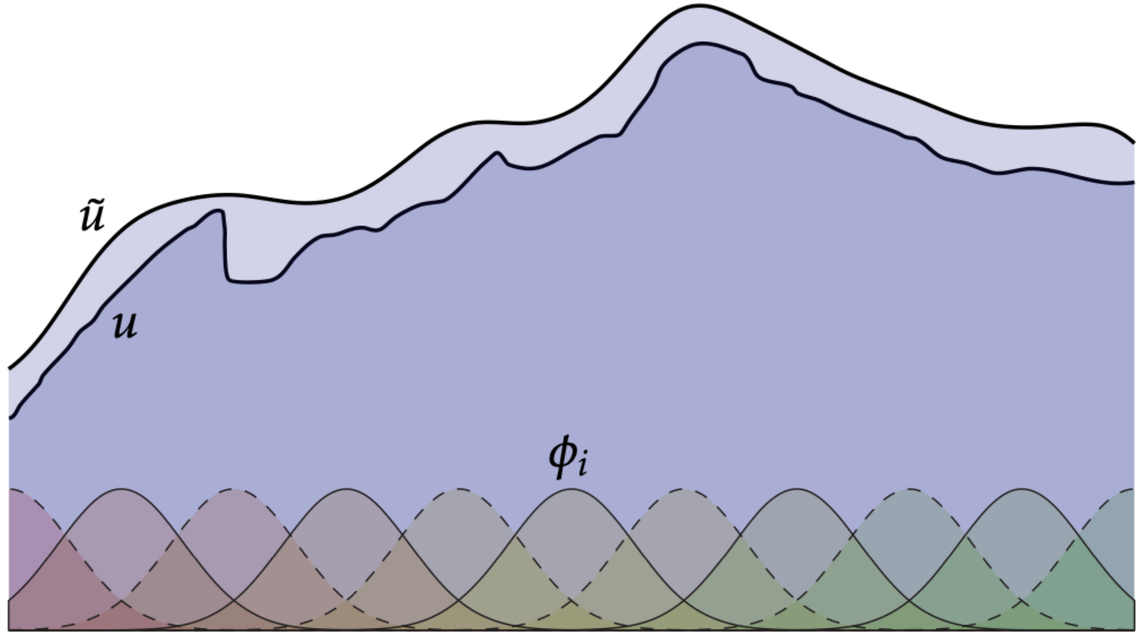
error due to approximation
of **geometry**

Traditional methods for solving PDEs

Inevitable consequence of finite dimensional approximation:



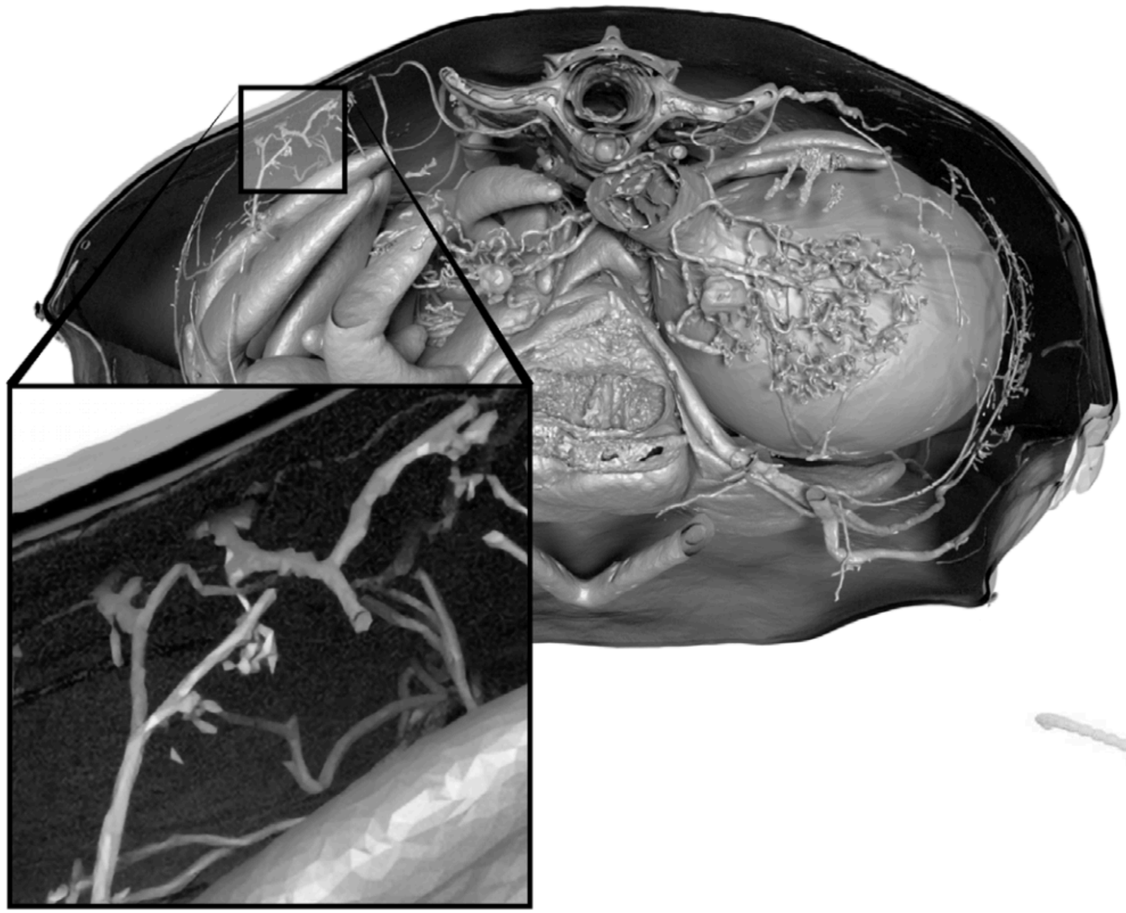
error due to approximation
of **geometry**

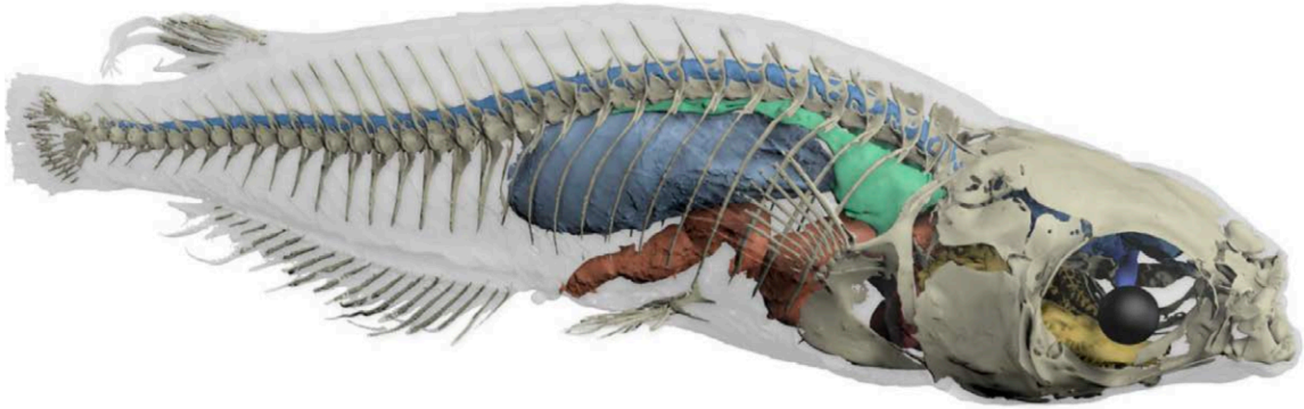


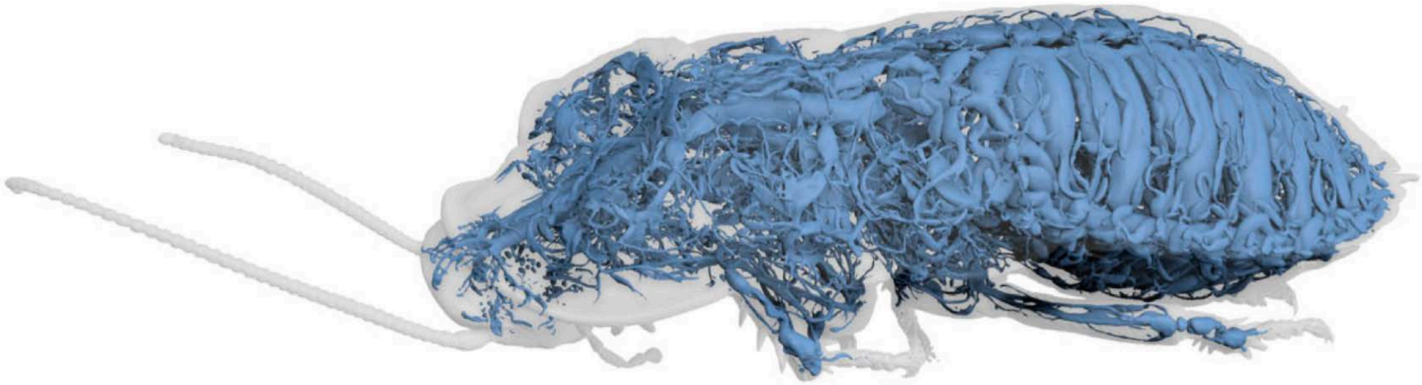
error due to approximation
of **functions**

Geometry found in physical world is extremely complex

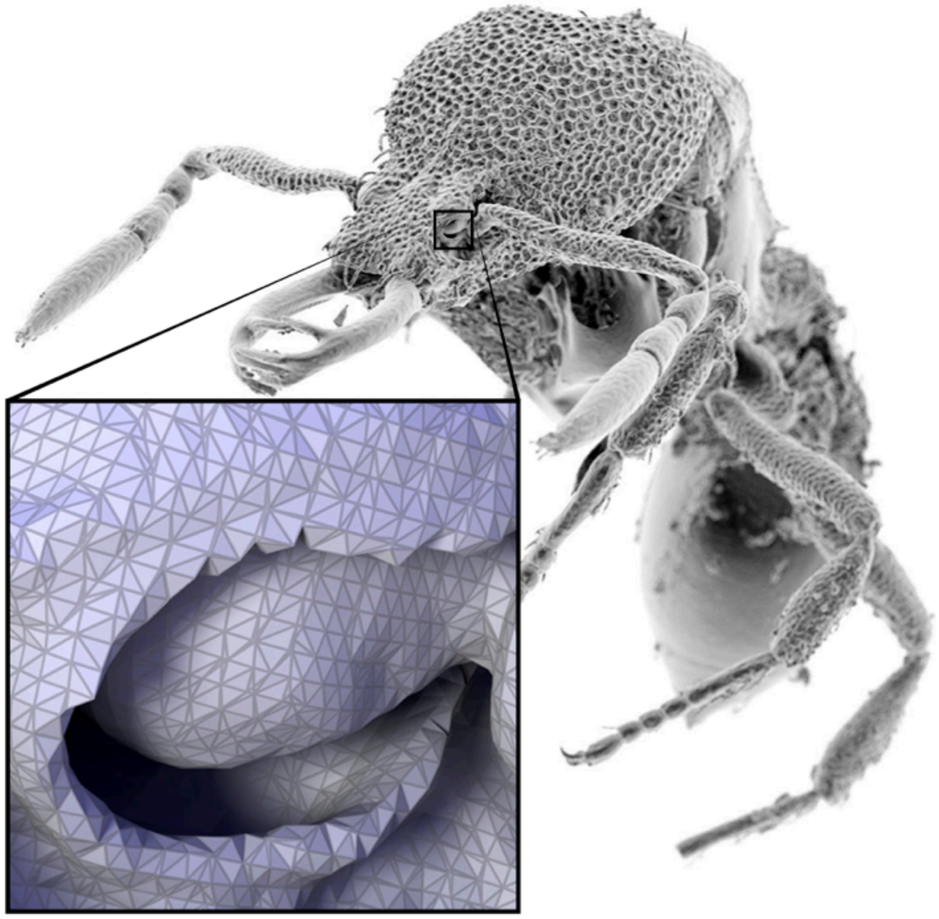
Example: high-resolution microCT scan







Complex geometry can take *extremely* long to mesh!

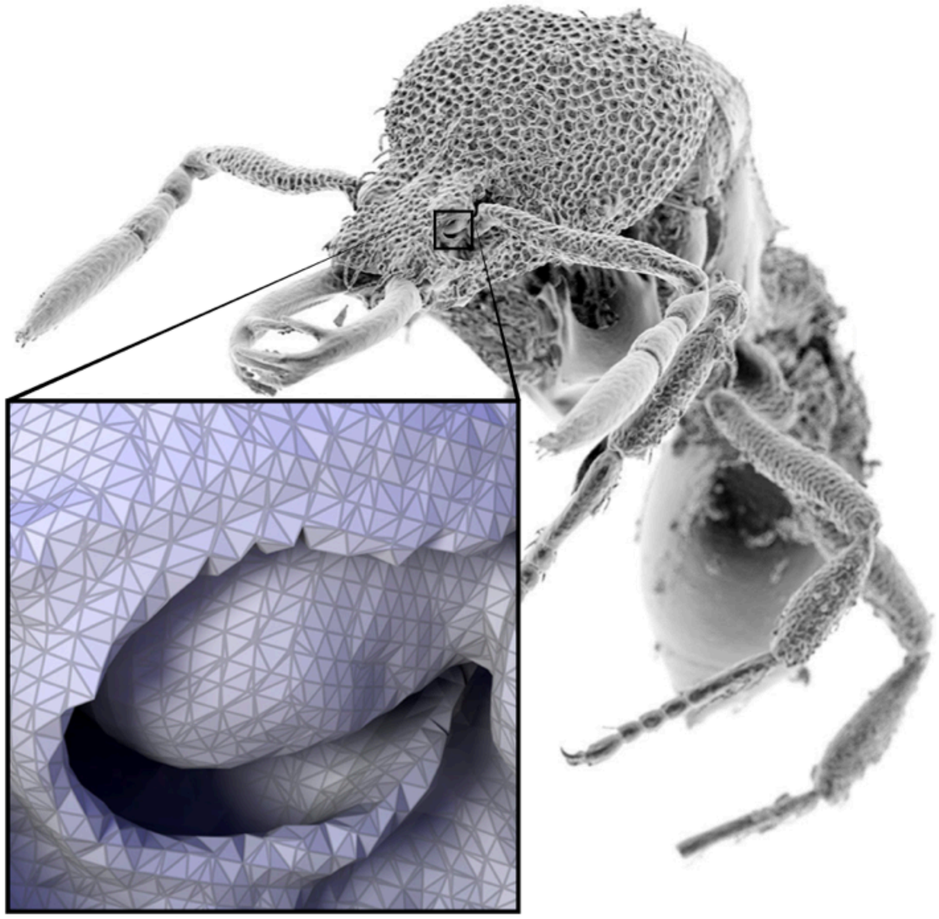


14 hours / 30 GB RAM

to generate "sim-ready" mesh

memory intensive & difficult to parallelize

Complex geometry can take *extremely* long to mesh!

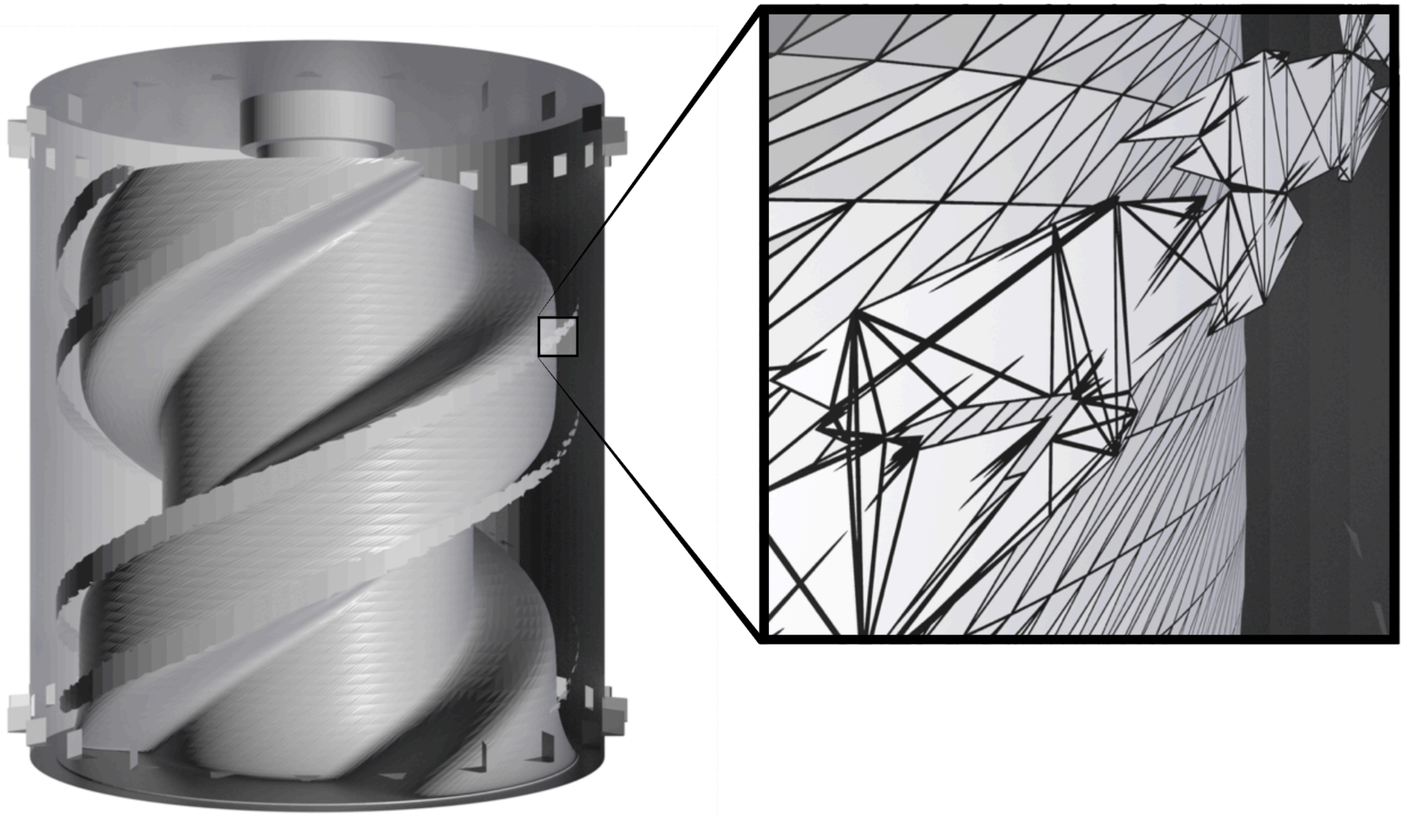


14 hours / 30 GB RAM

to generate "sim-ready" mesh

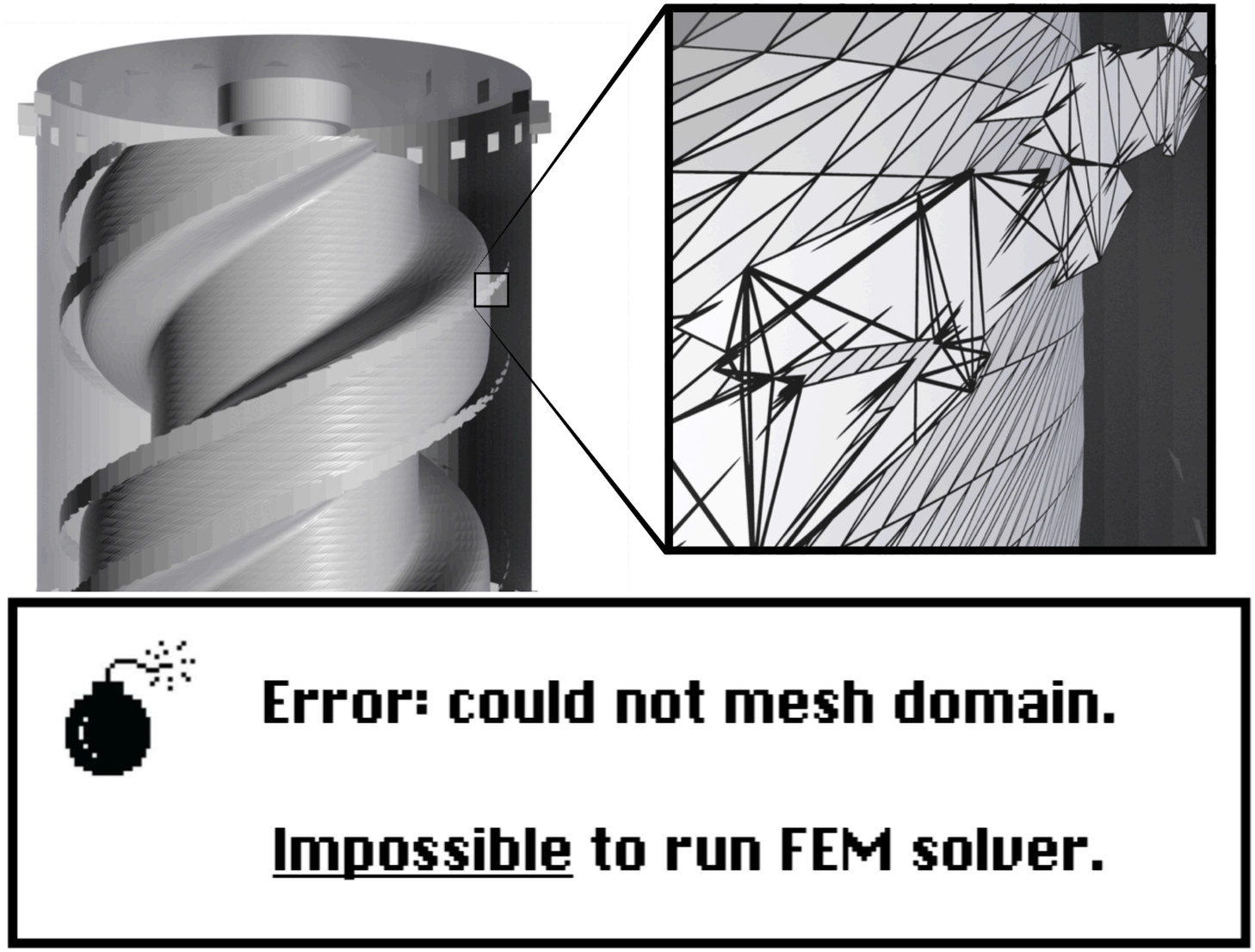
memory intensive & difficult to parallelize

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



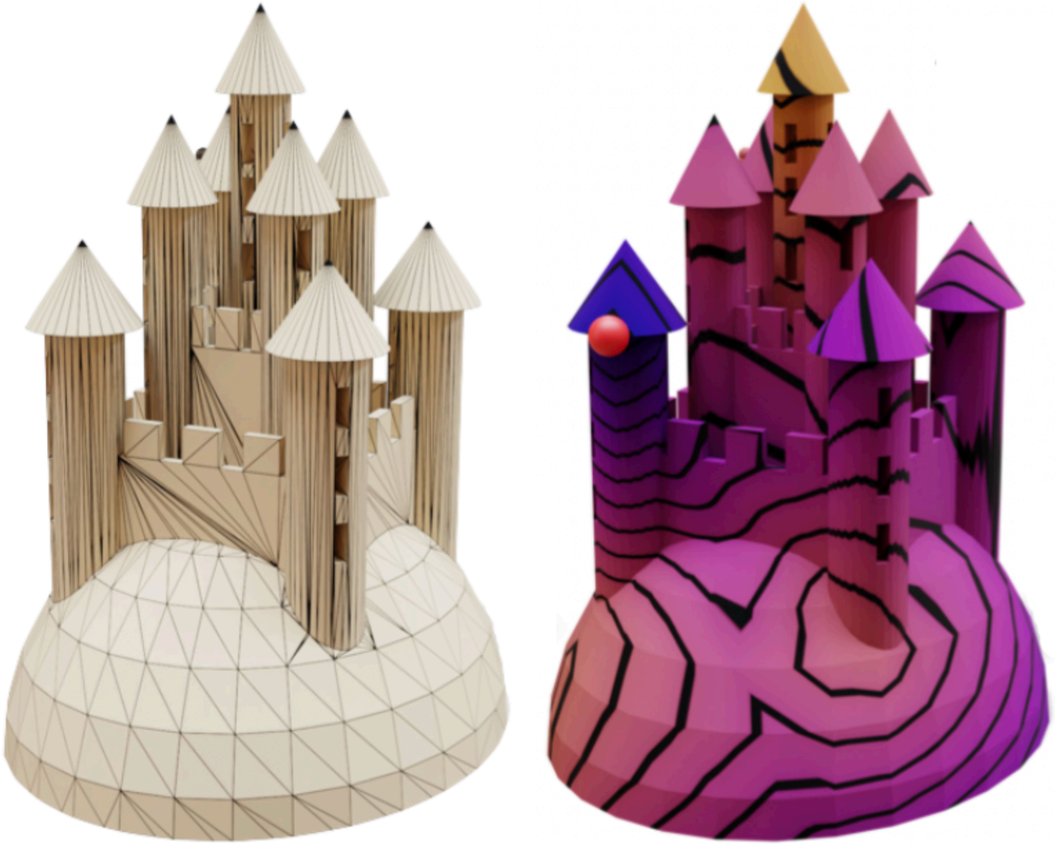
"Hell is other people's meshes."
—Jean-Paul Sartre

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



A bad mesh can yield a false impression of reality

Example: geodesic distance via FEM



FEM solution

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

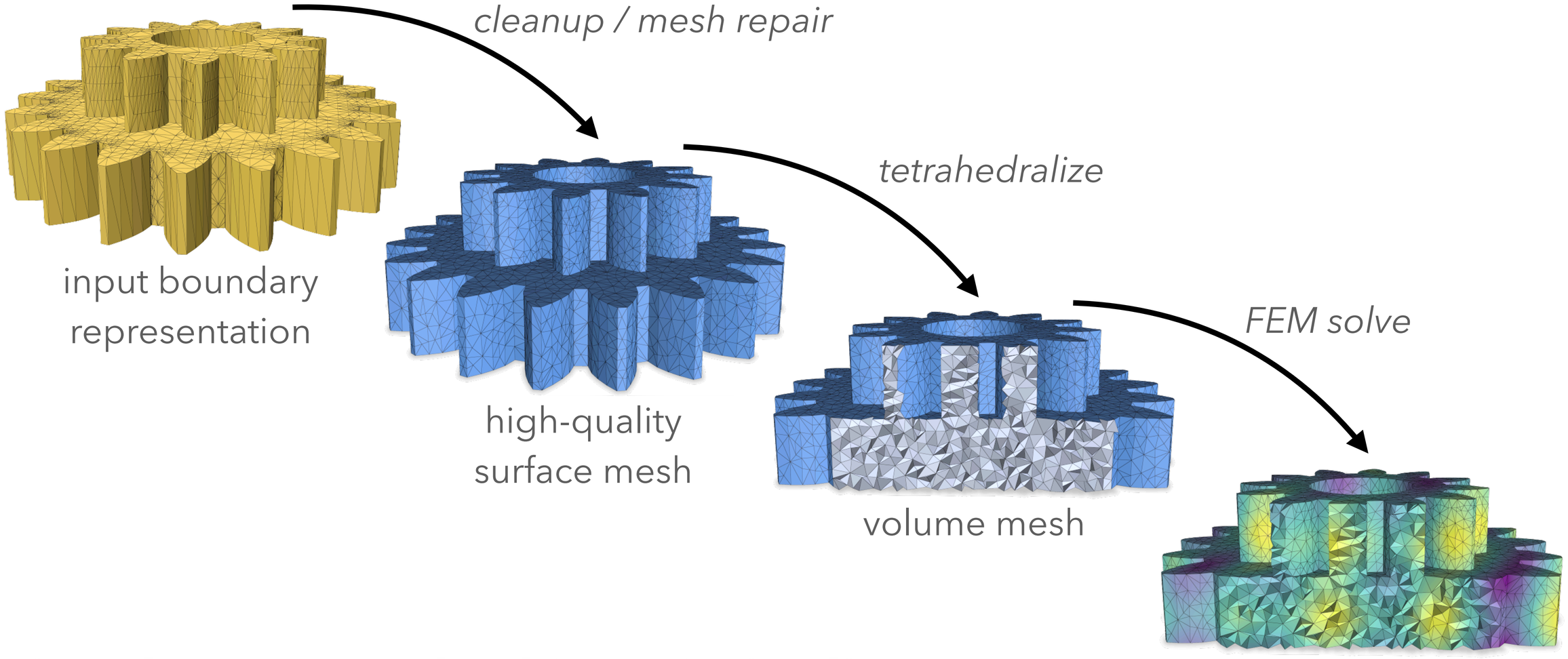
A bad mesh can yield a false impression of reality

Example: geodesic distance via FEM

FEM solution reference solution

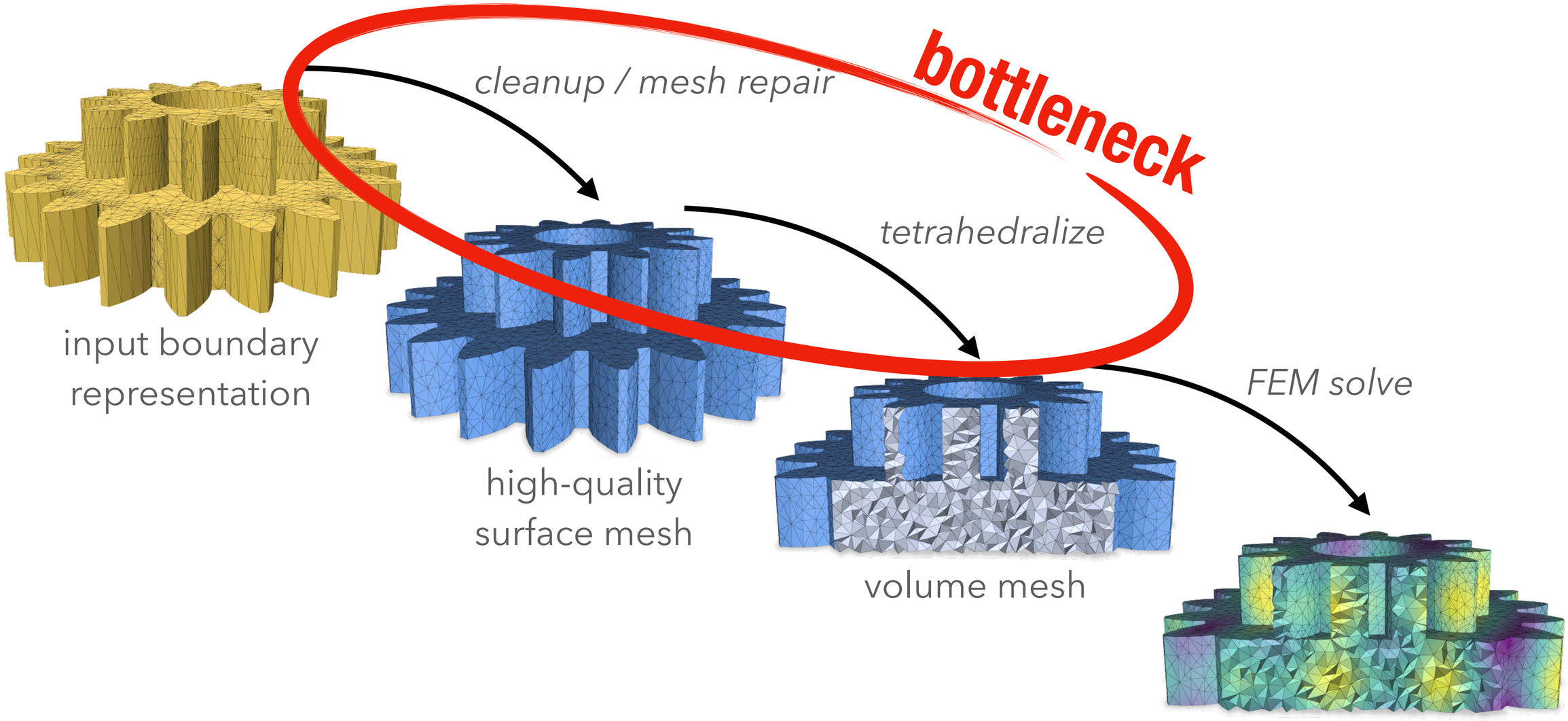
N. Sharp & K. Crane, A Laplacian for Nonmanifold Meshes (2020)

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



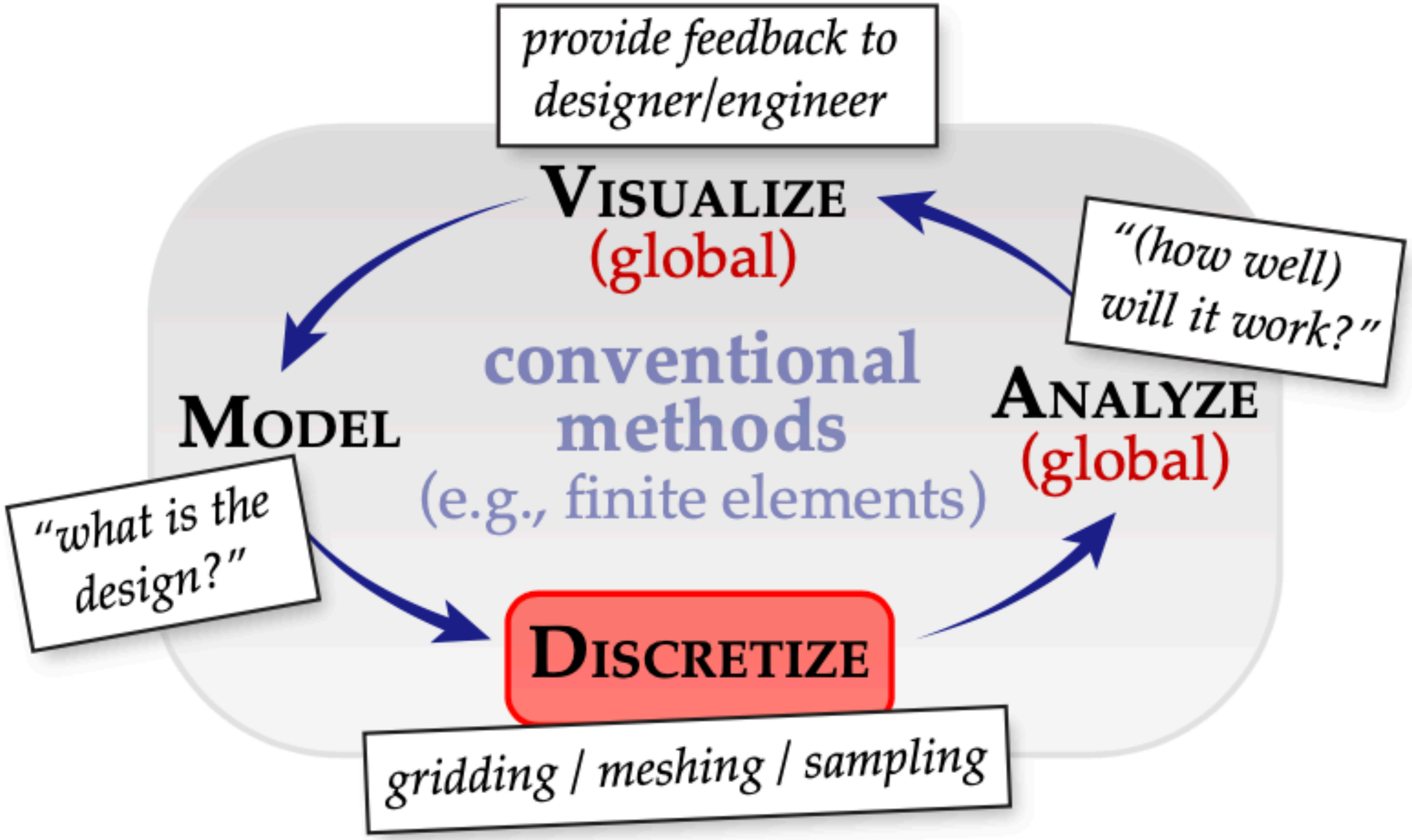
finite element method (FEM) pipeline

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



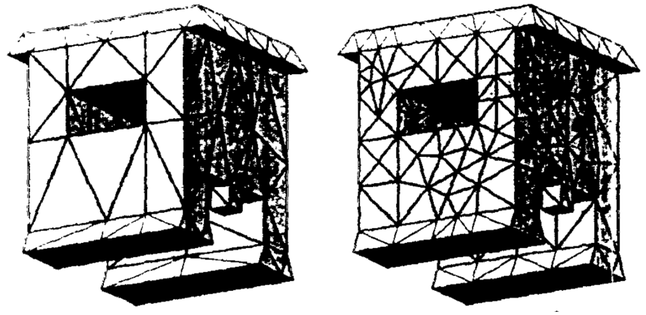
finite element method (FEM) pipeline

Meshing is *always* the bottleneck for simulation!

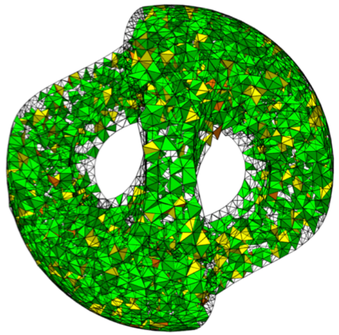


Robust meshing is still hard, even after 20+ years

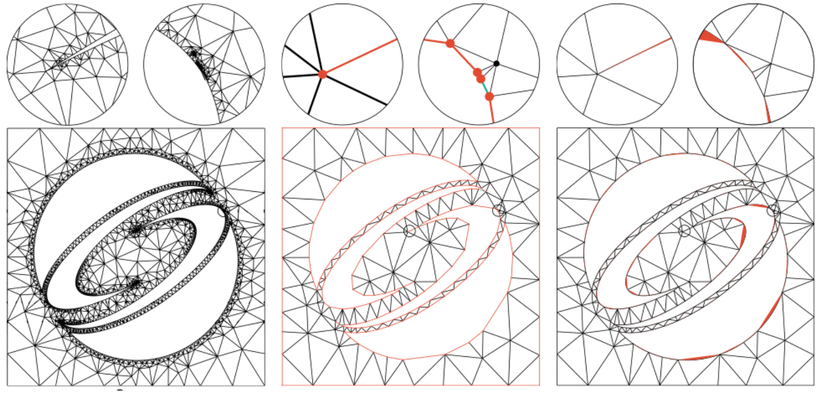
Not likely to **ever** be completely "solved":



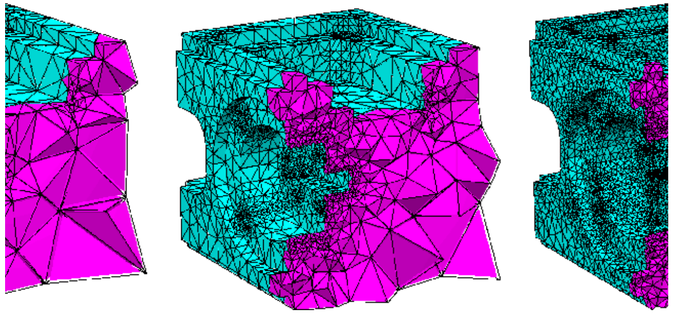
Tetrahedral Mesh Generation by Delaunay Refinement **[Shewchuk 1998]**



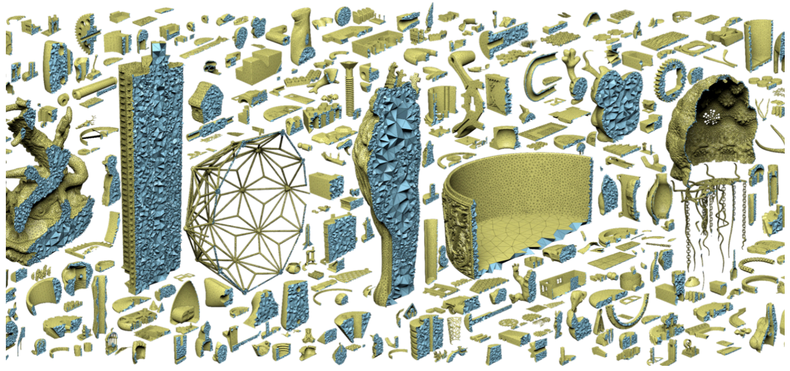
Robust Tetrahedral Meshing of Triangle Soups **[Spillman et al. 2006]**



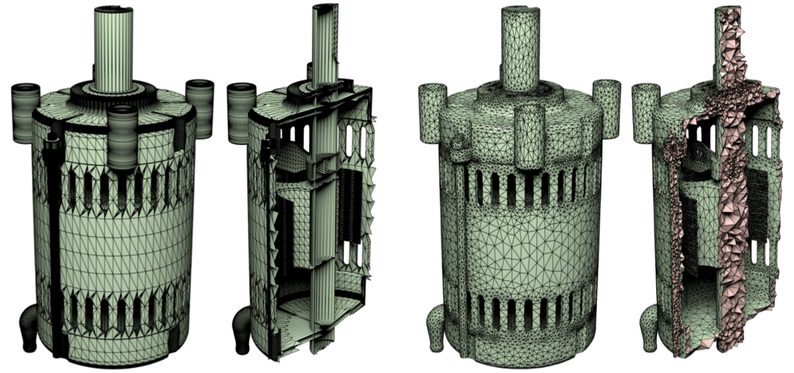
TriWild: Robust Triangulation With Curve Constraints **[Hu et al. 2019]**



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

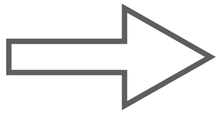


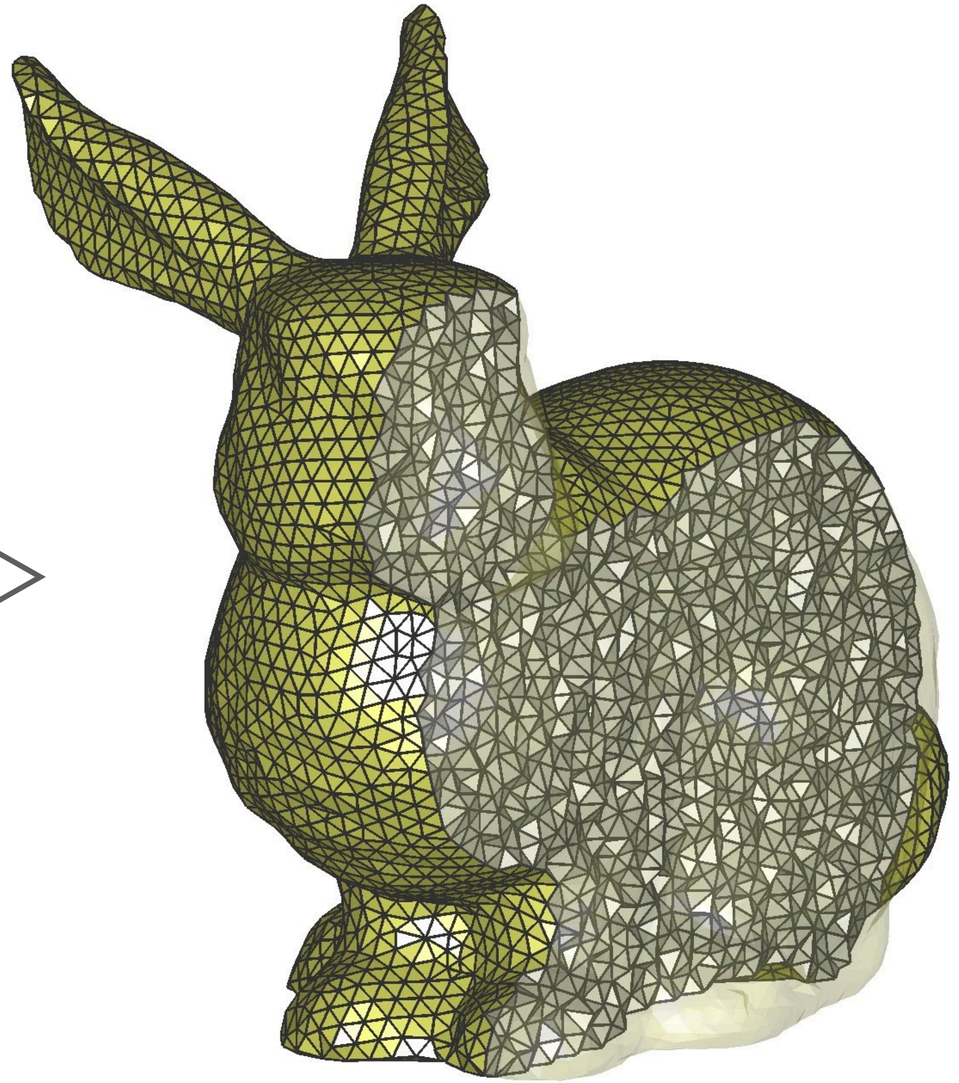
Tetrahedral Meshing in the Wild **[Hu et al. 2018]**



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

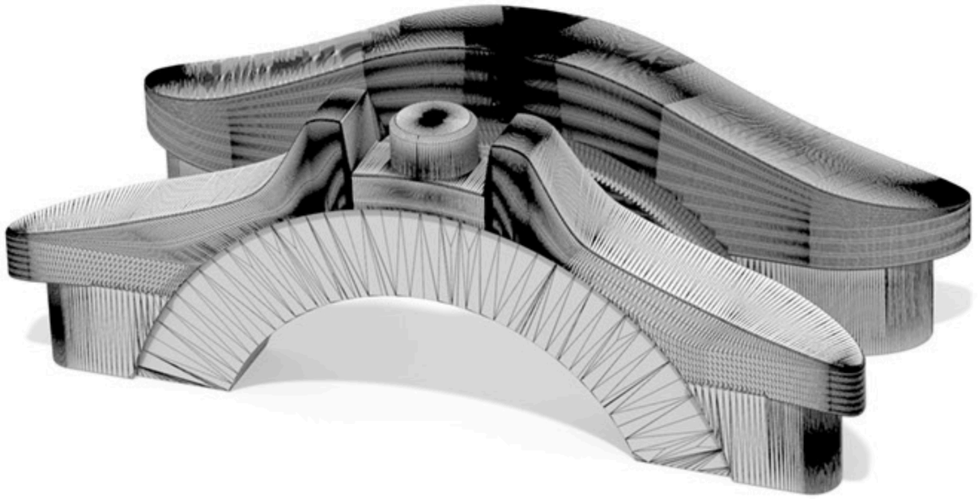
Traditional PDE solvers require volumetric meshing



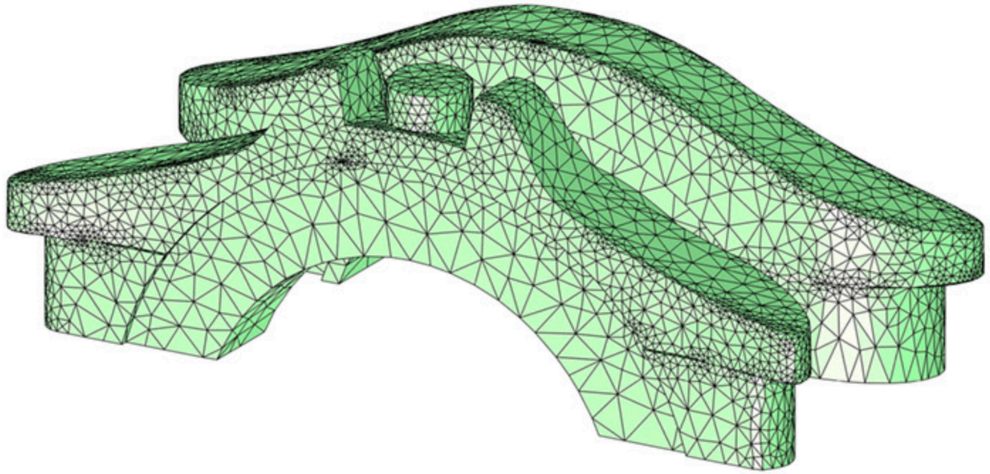


Robust meshing can be *wildly* unpredictable

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



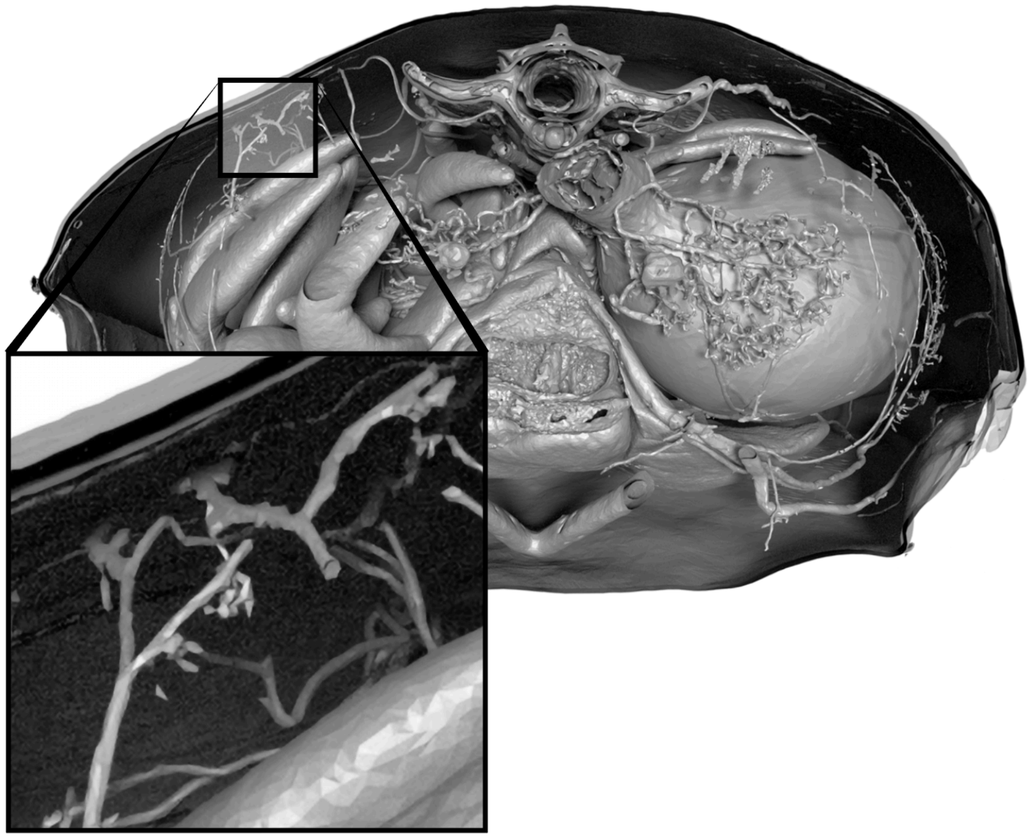
Input (Thingi10k #996816)



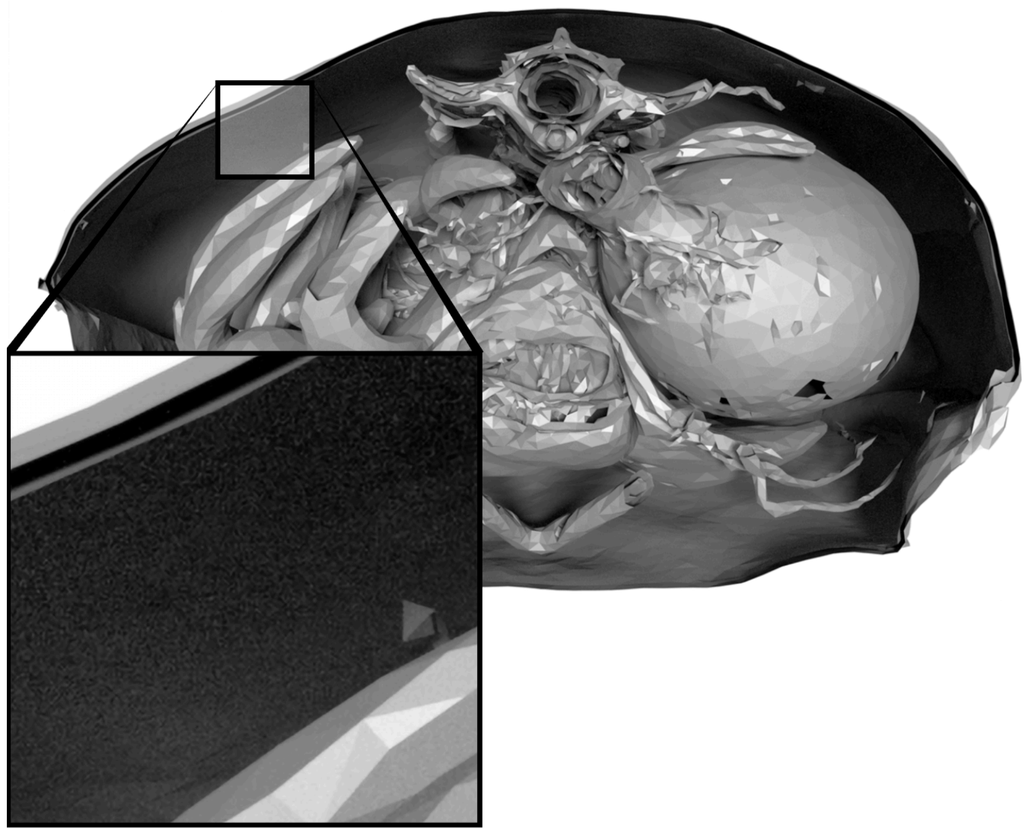
FastTetWild, **1 hour 25 minutes**

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

Even when meshing “succeeds”, critical details can be lost

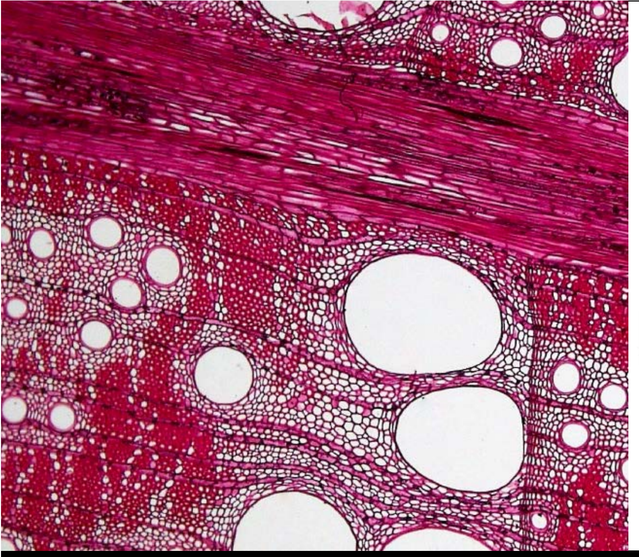


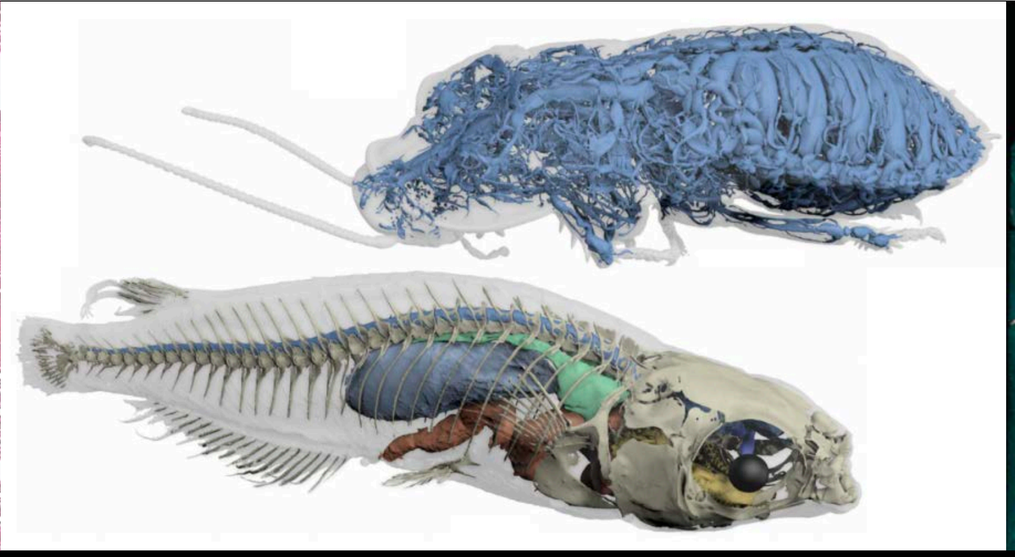
boundary mesh (*input*)



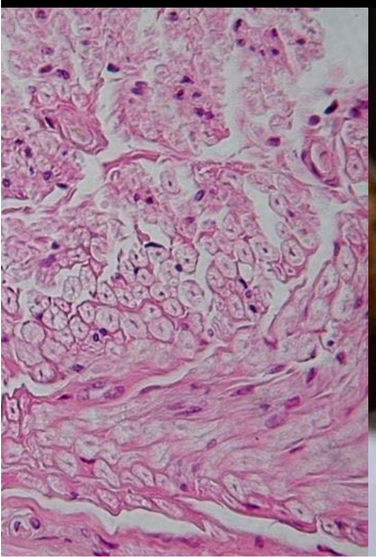
FastTetWild (*output boundary*)

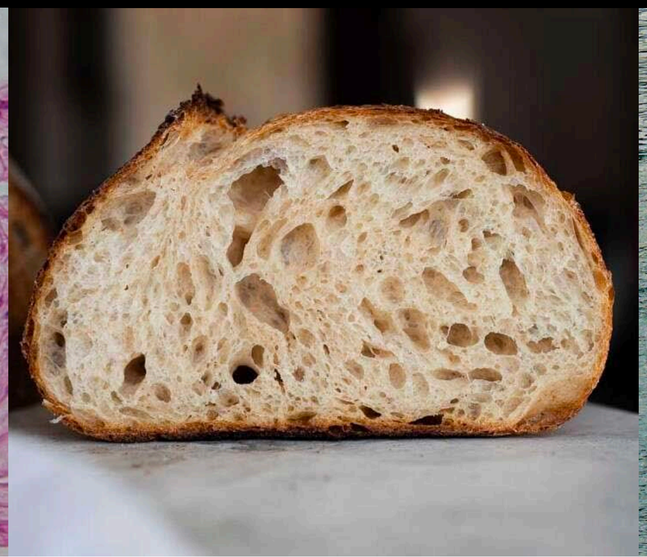
Hu et al, *“Fast Tetrahedral Meshing in the Wild”* (2020)

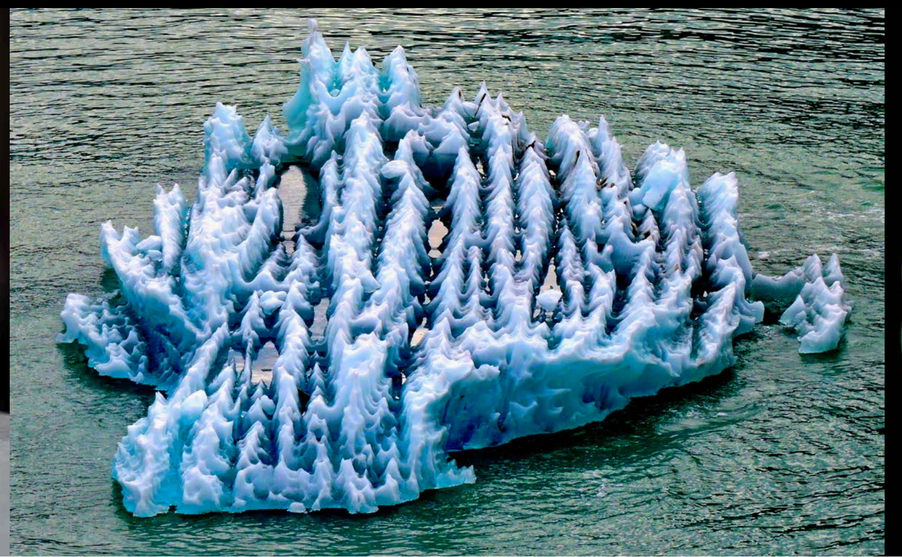




To faithfully simulate nature, we *must* be able to handle a much greater level of **geometric complexity**.







Rendering: from Finite Elements to Monte Carlo

Early days of rendering: *finite element radiosity*

Rendering: from Finite Elements to Monte Carlo

Early days of rendering: *finite element radiosity*

mesh scene

Rendering: from Finite Elements to Monte Carlo

Early days of rendering: *finite element radiosity*

mesh scene

↓

setup large matrix

Rendering: from Finite Elements to Monte Carlo

Early days of rendering: *finite element radiosity*

mesh scene

↓

setup large matrix

↓

perform global solve

Rendering: from Finite Elements to Monte Carlo

Early days of rendering: *finite element radiosity*

global & painful!

mesh scene

↓

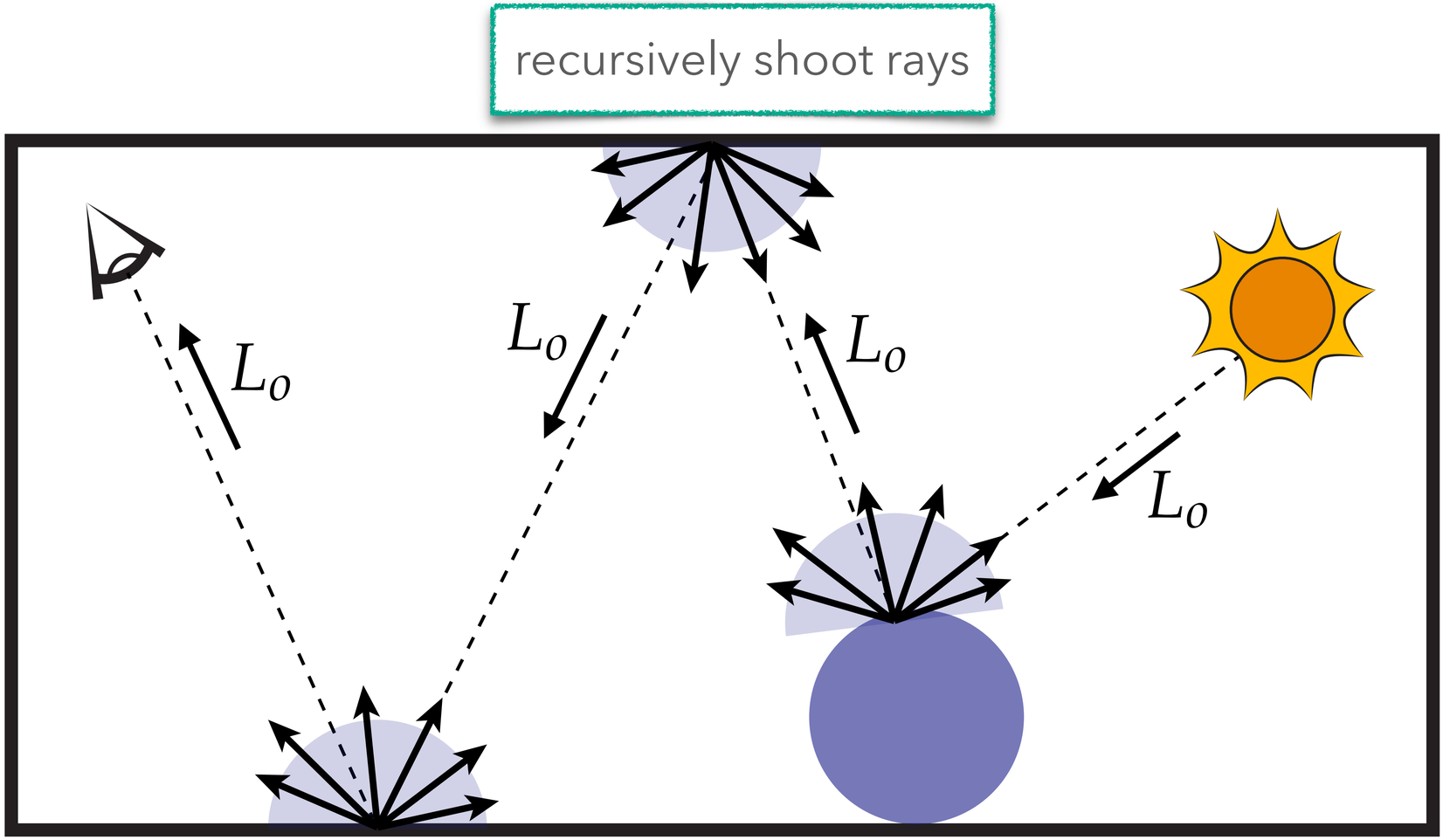
setup large matrix

↓

perform global solve

Rendering: from Finite Elements to Monte Carlo

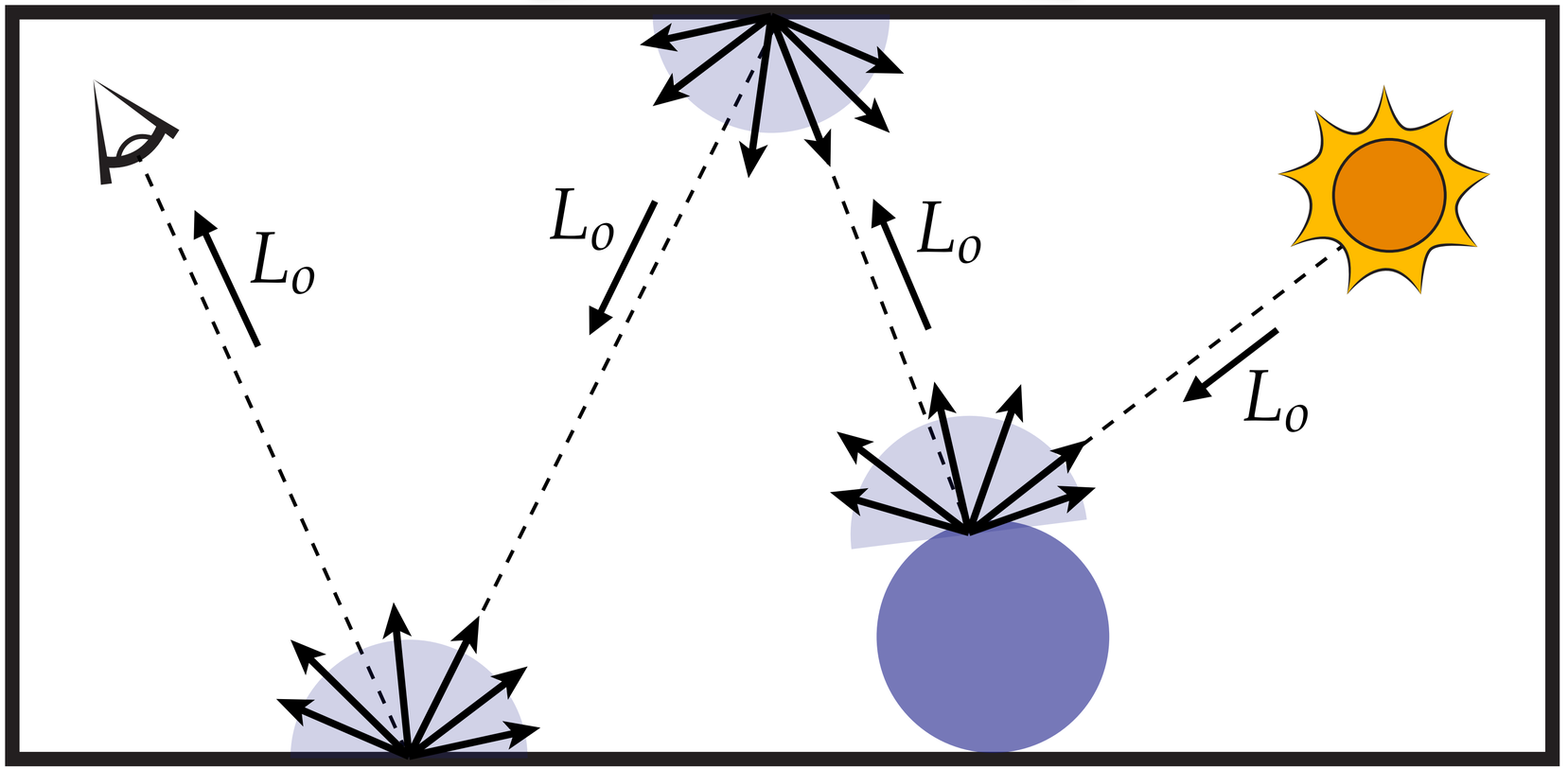
Monte Carlo ray tracing avoids meshing **entirely**, via repeated random sampling



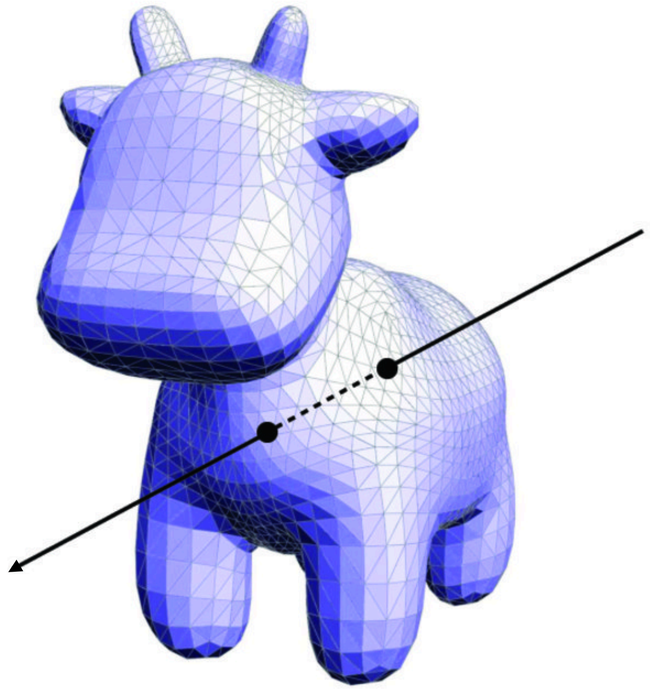
Rendering: from Finite Elements to Monte Carlo

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

recursively shoot rays



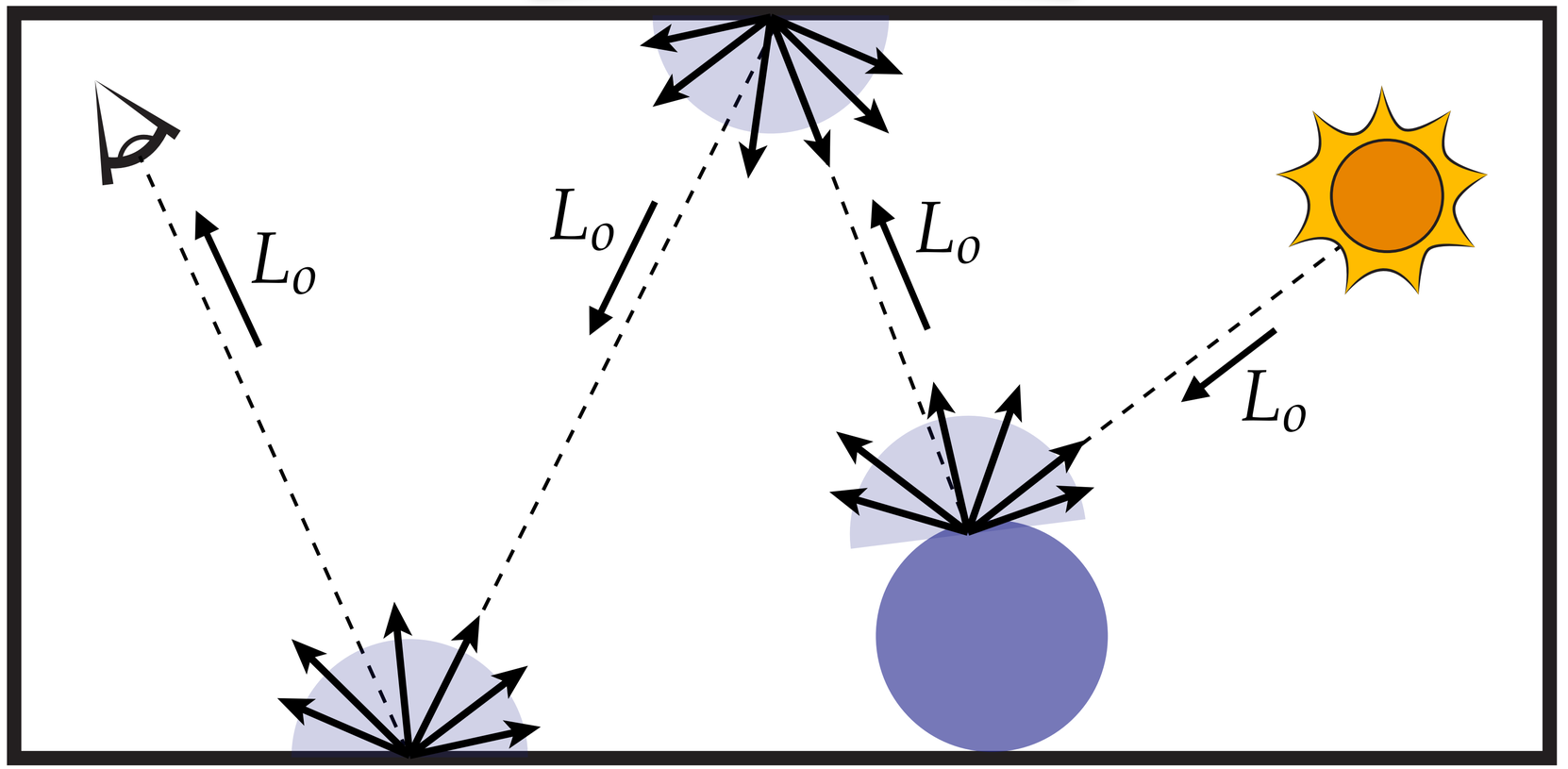
ray-scene intersections



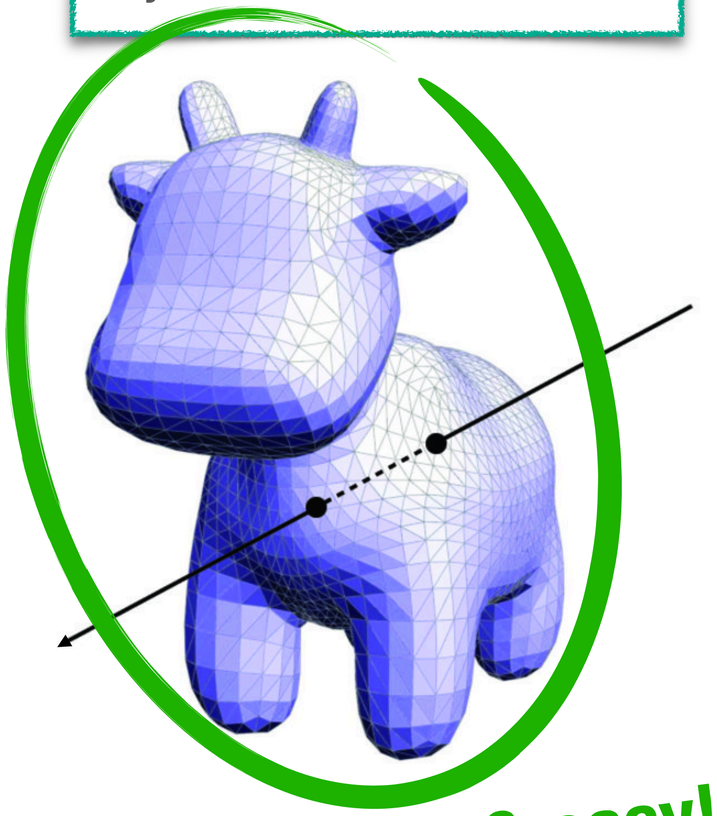
Rendering: from Finite Elements to Monte Carlo

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

recursively shoot rays



ray-scene intersections



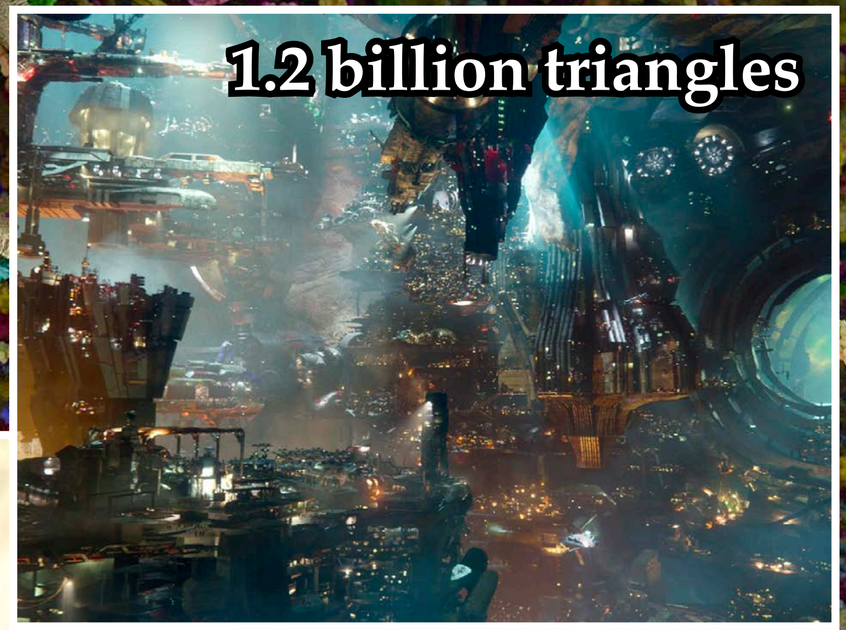
local & easy!

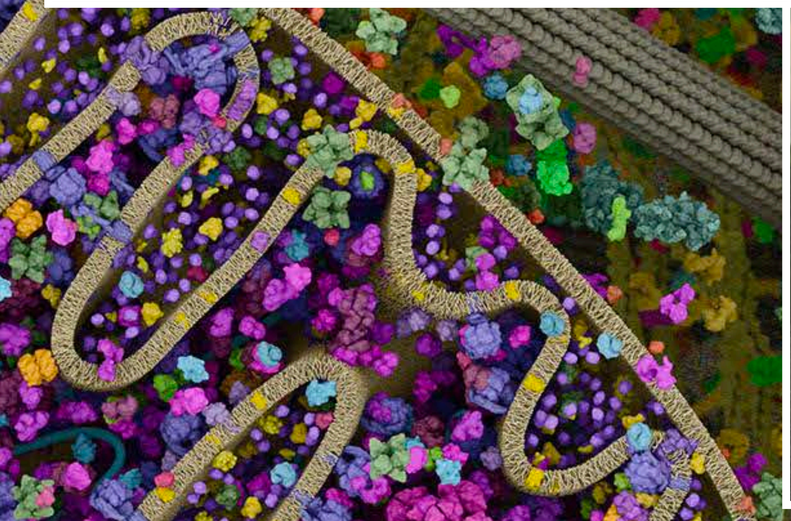
Monte Carlo methods

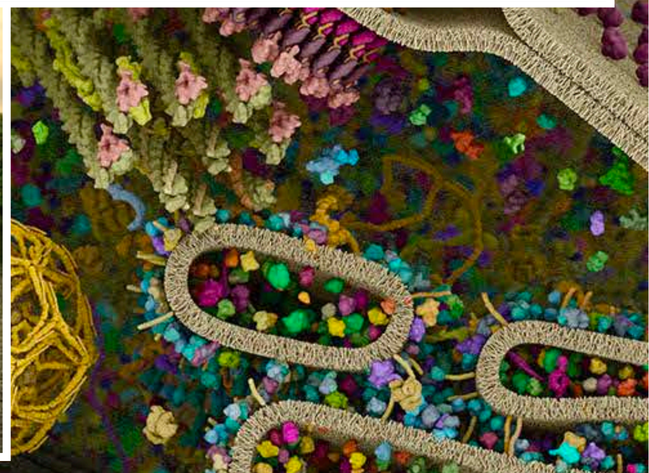
Evaluate an integral by averaging its integrand N times:

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

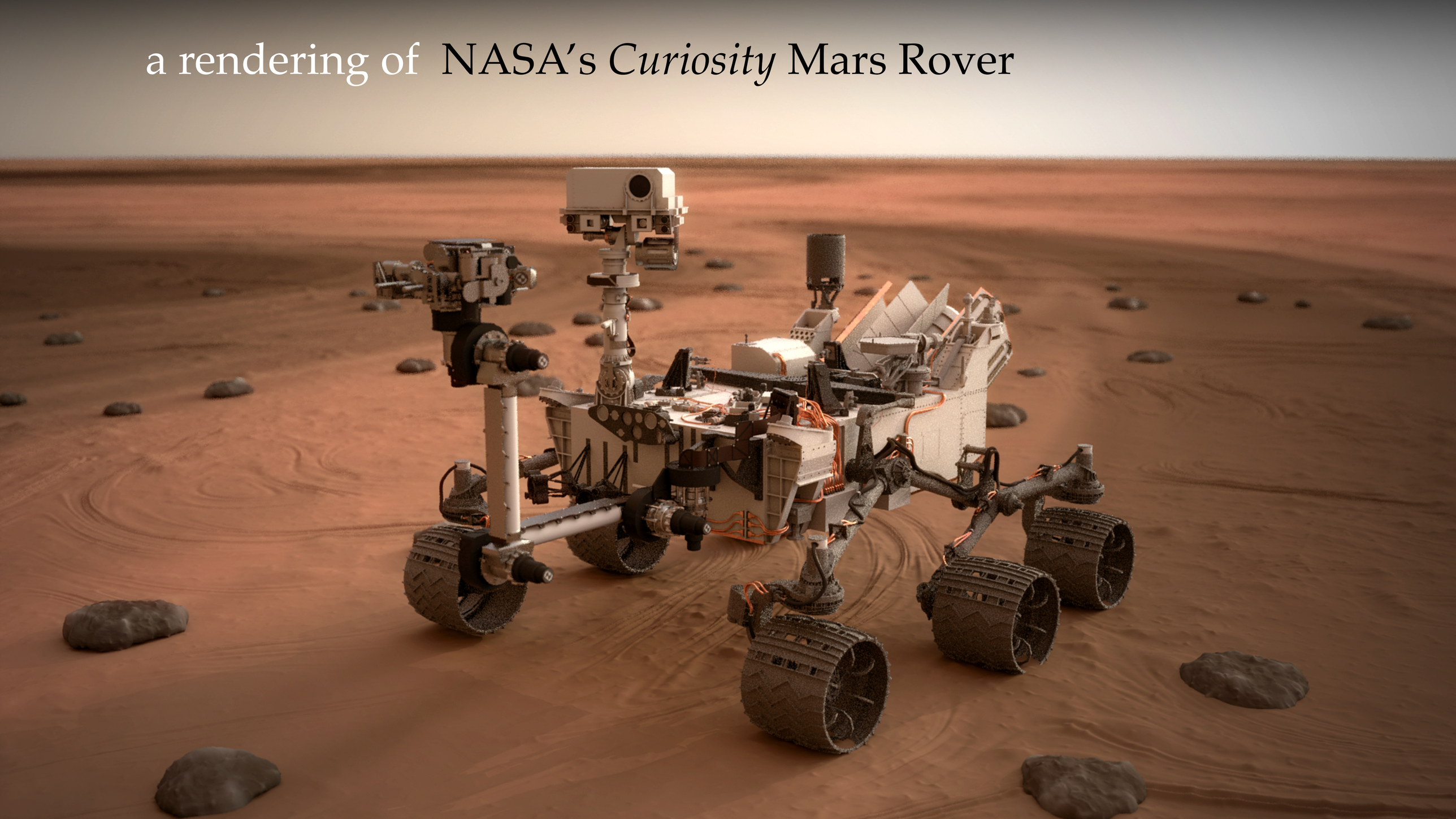
*Monte Carlo rendering can now handle **immense** geometric complexity*



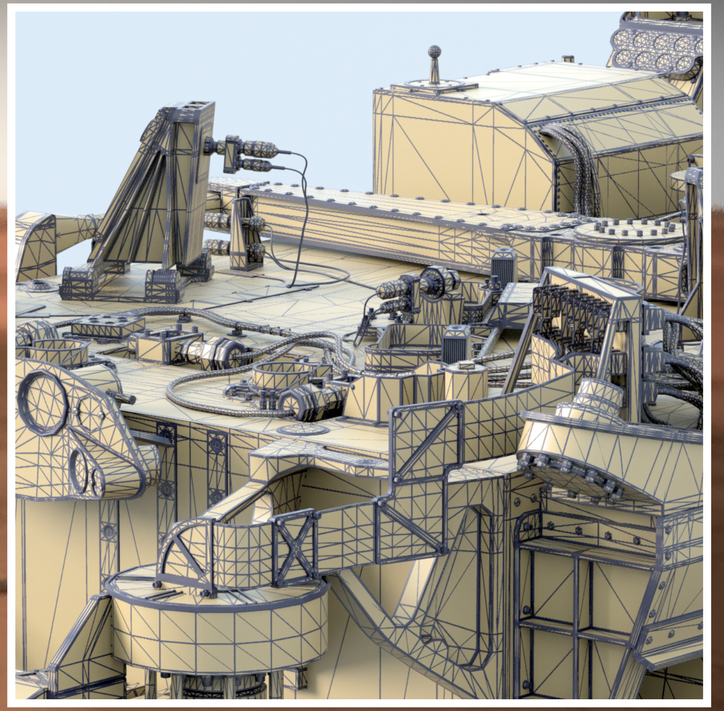


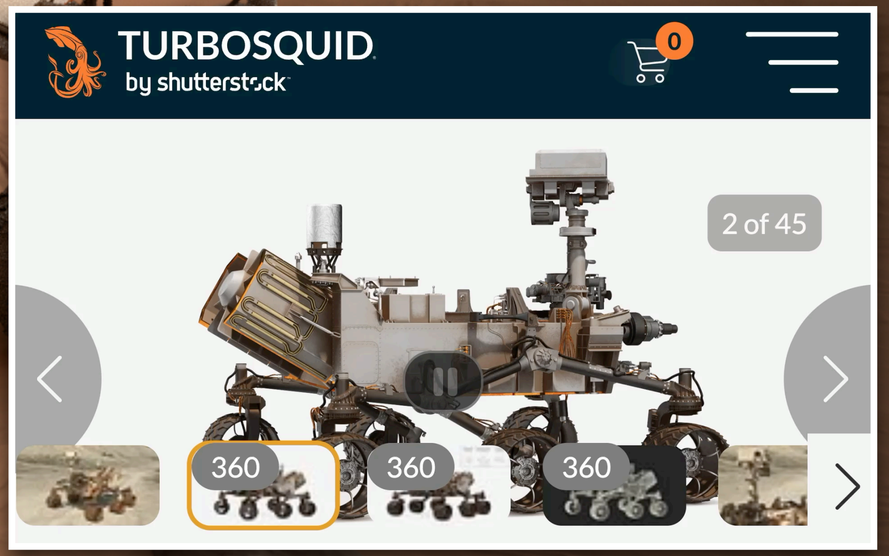


a rendering of NASA's *Curiosity* Mars Rover

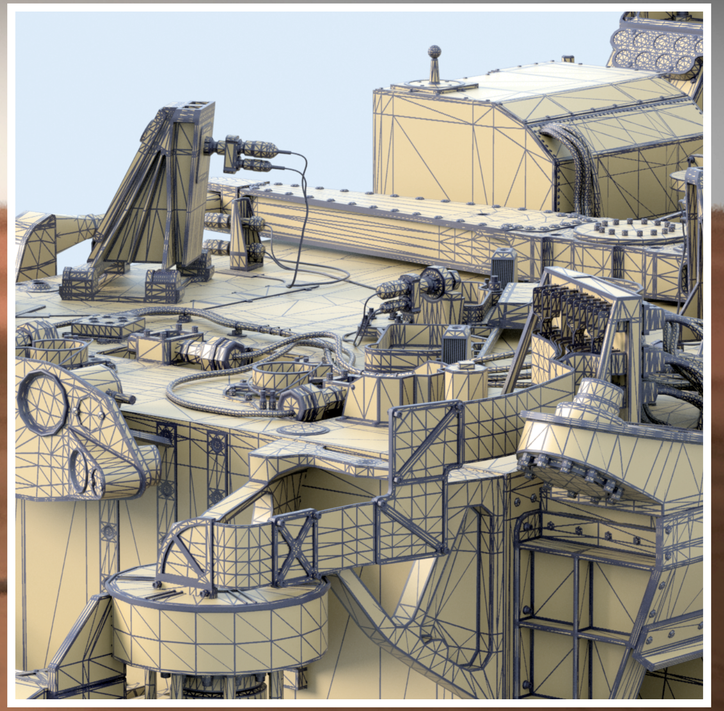


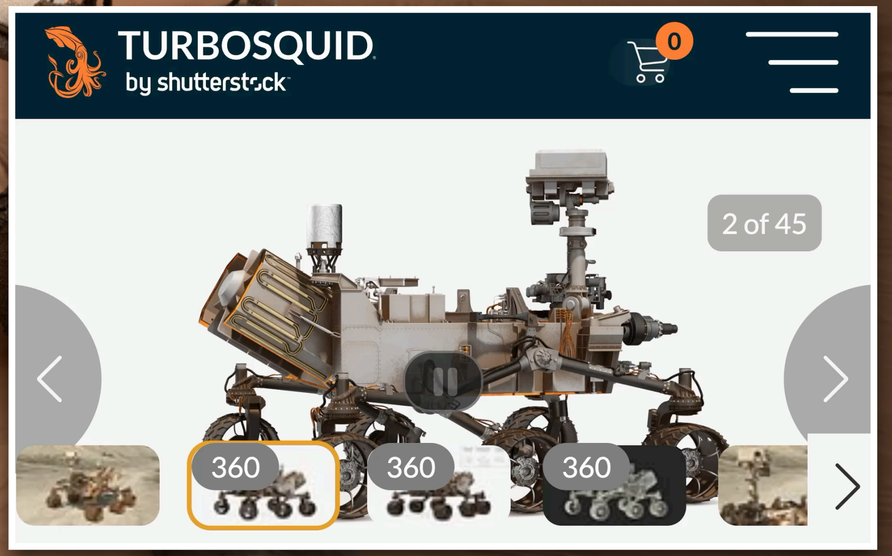
a rendering of NASA's *Curiosity* Mars Rover





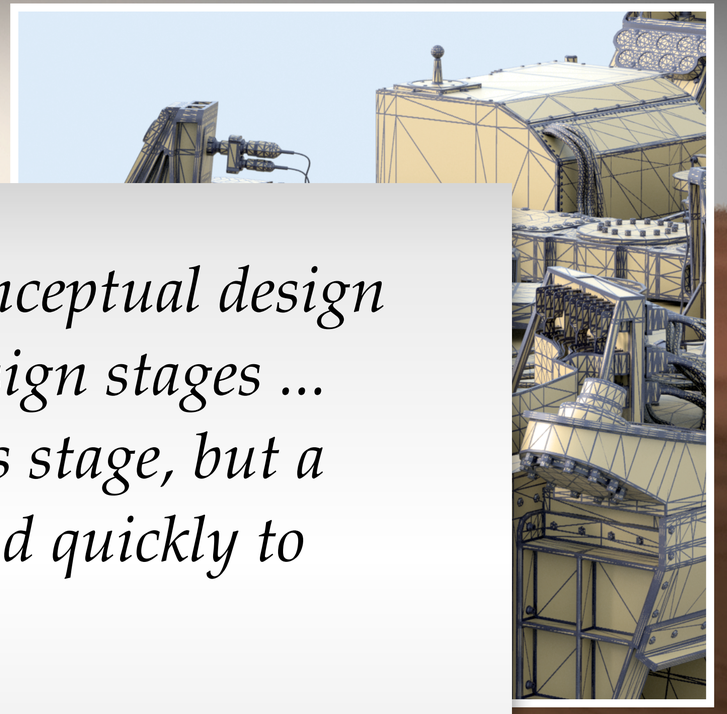
a rendering of NASA's *Curiosity* Mars Rover





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

a rendering of NASA's *Curiosity* Mars Rover

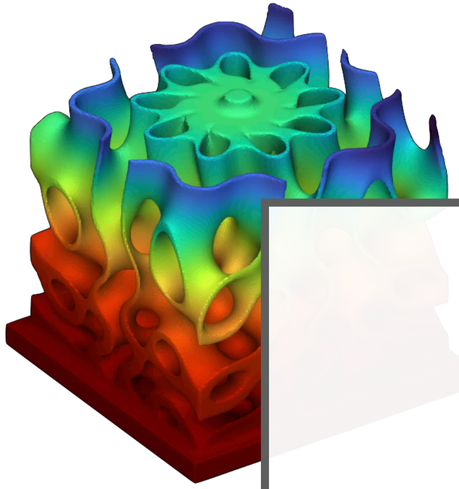


"Thermal modeling is required beginning at the project conceptual design stage and continuing through preliminary and detailed design stages ... simplified calculations and rules of thumb are useful at this stage, but a computer model provides the ability to evaluate and respond quickly to proposed system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

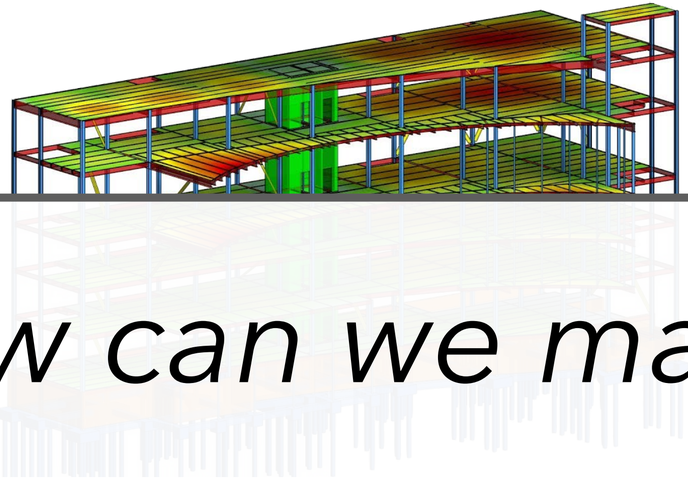
—NASA Guidelines for Thermal Analysis of Spacecraft Hardware

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

Physics beyond light transport



thermal diffusion



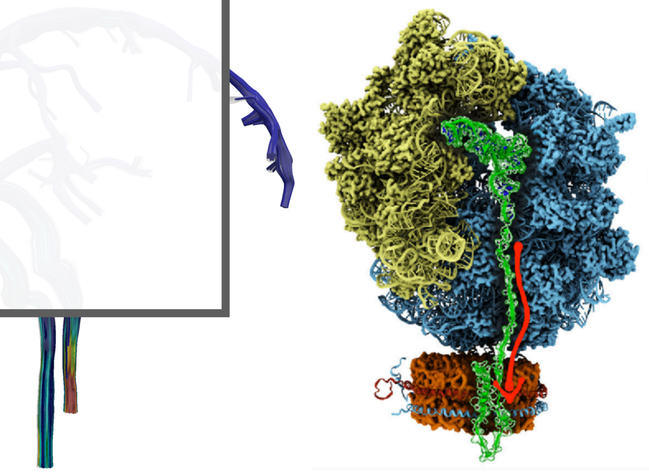
structural analysis

electrostatics

How can we make simulation more like rendering?

acoustic modeling

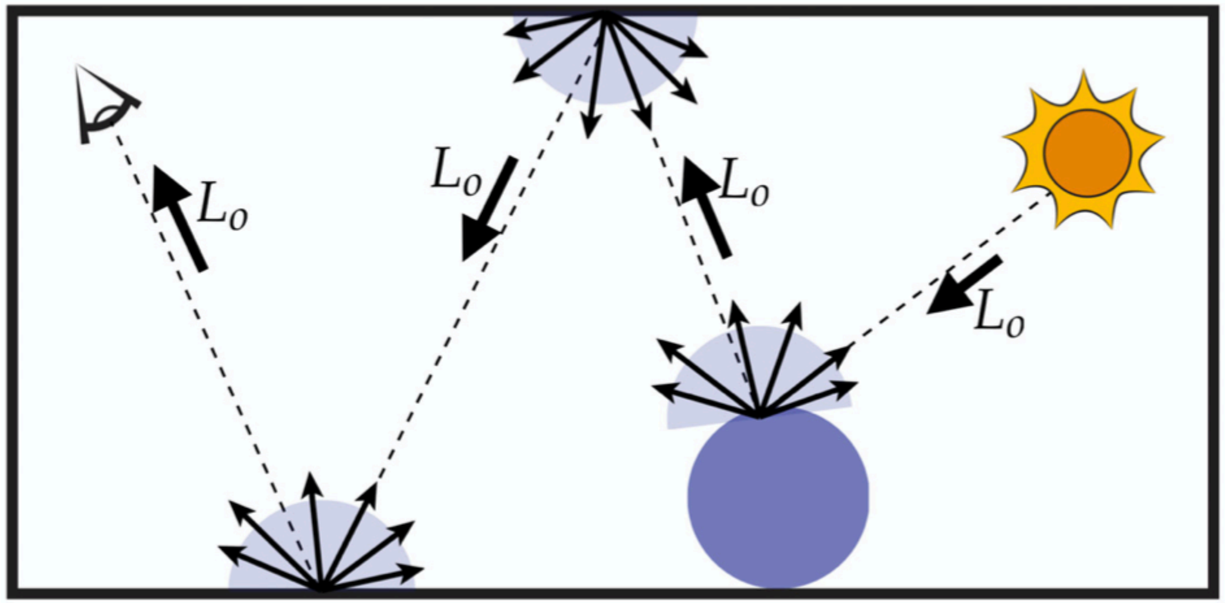
microfluidics



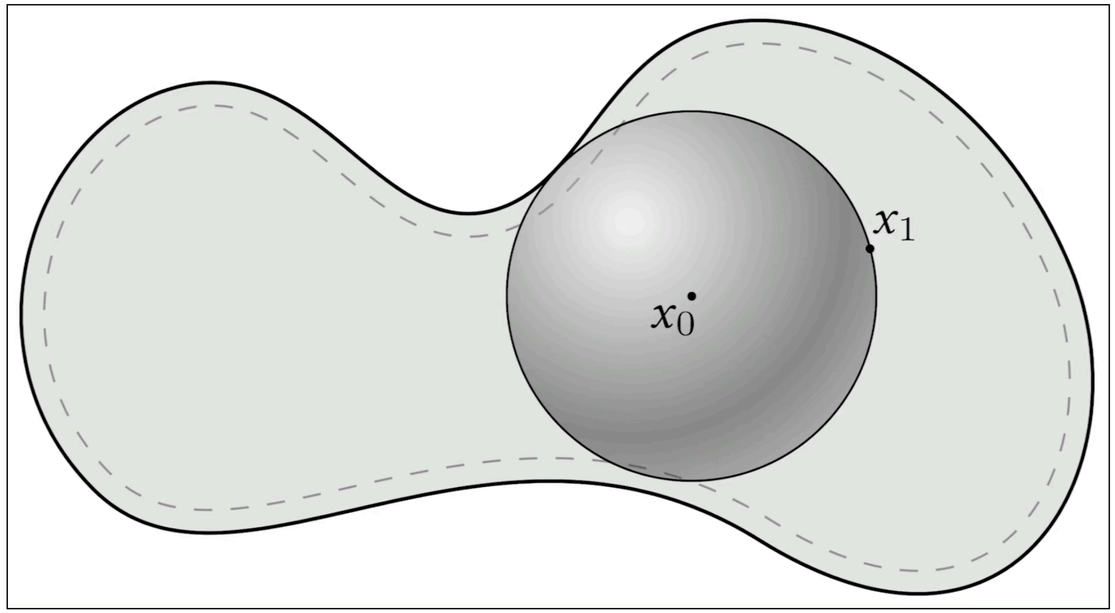
biophysics

From Ray Tracing to Random Walks

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



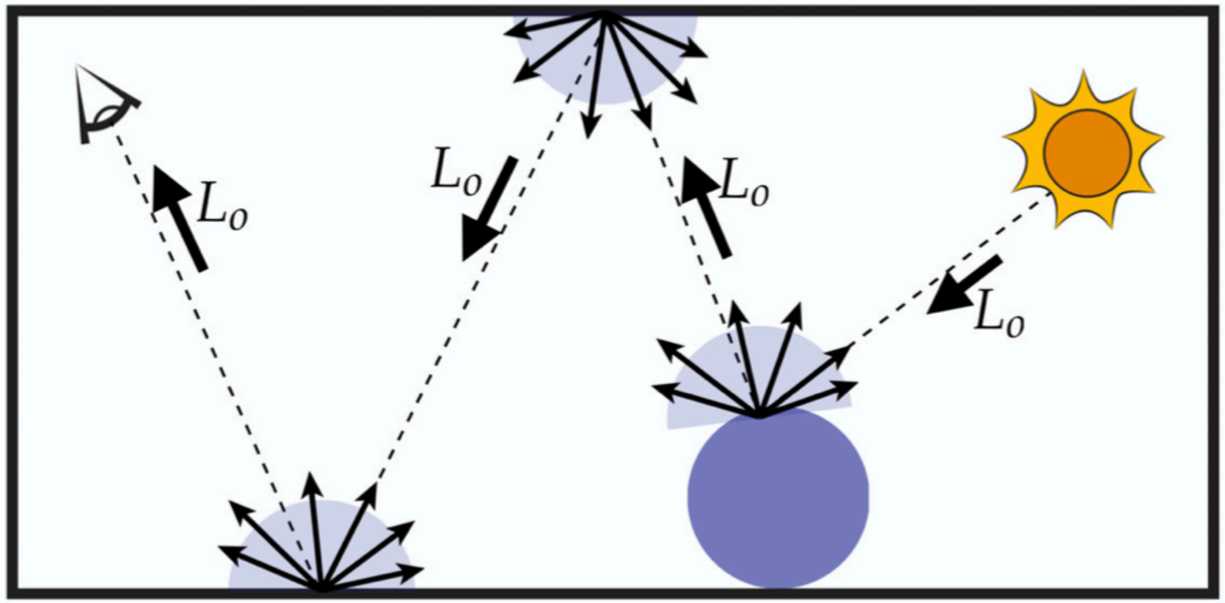
ray tracing



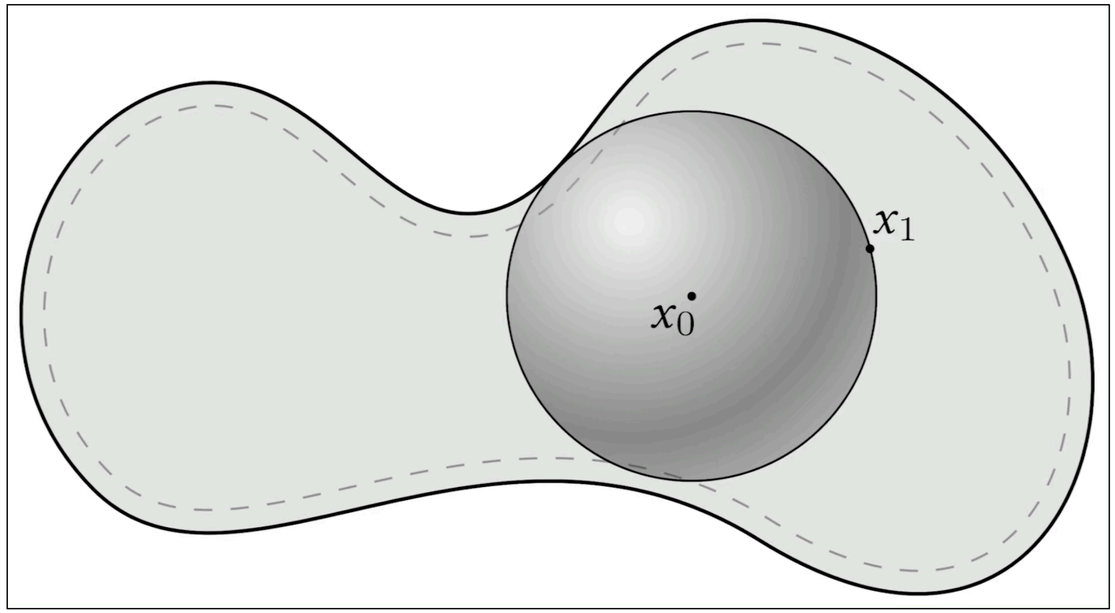
walk on spheres [Muller 1956]

From Ray Tracing to Random Walks

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

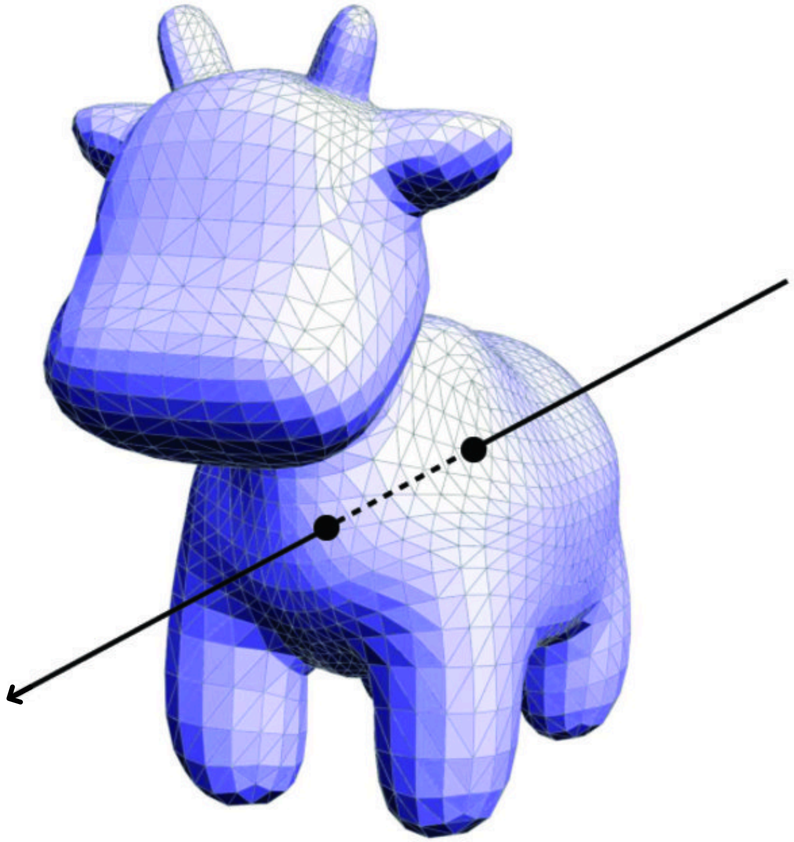


ray tracing

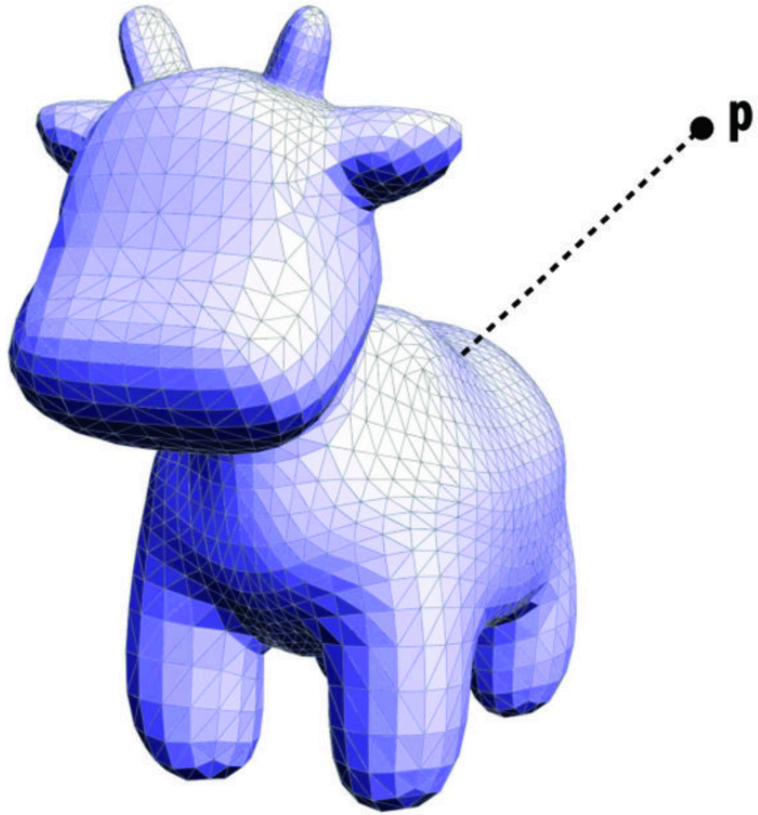


walk on spheres [Muller 1956]

From Ray Intersections to Closest Point Queries

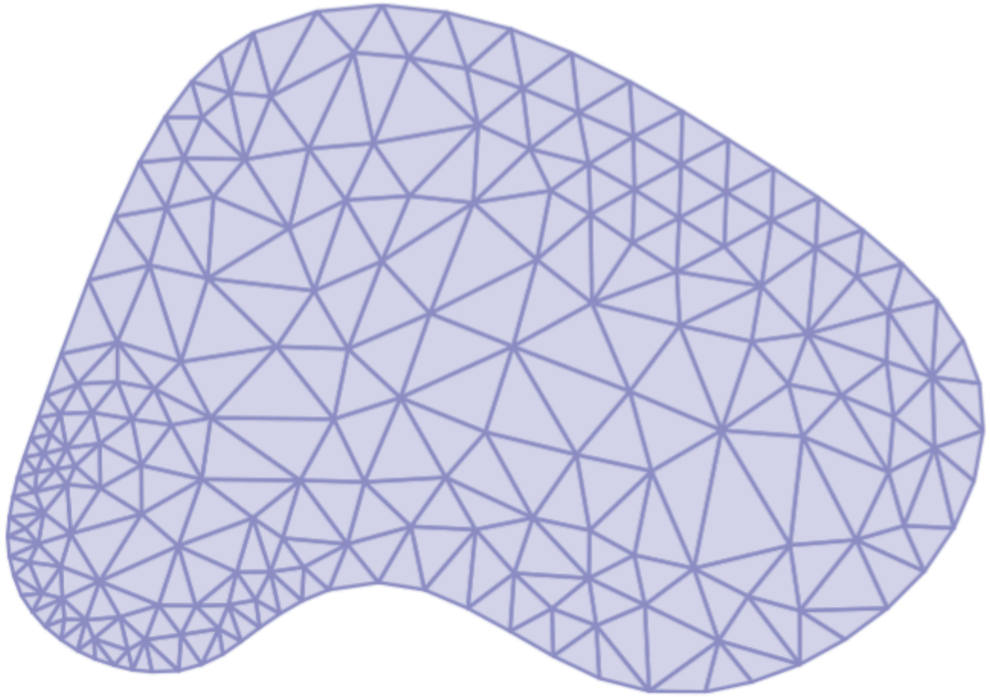


Ray Intersection Query

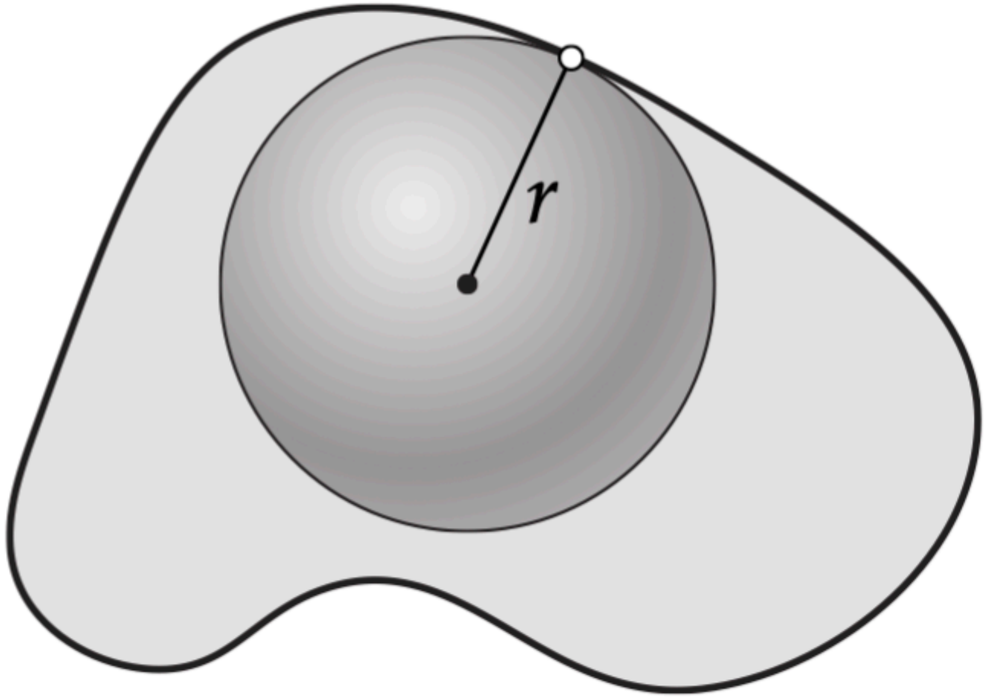


Closest Point Query

Meshing is hard…finding closest point is easy!

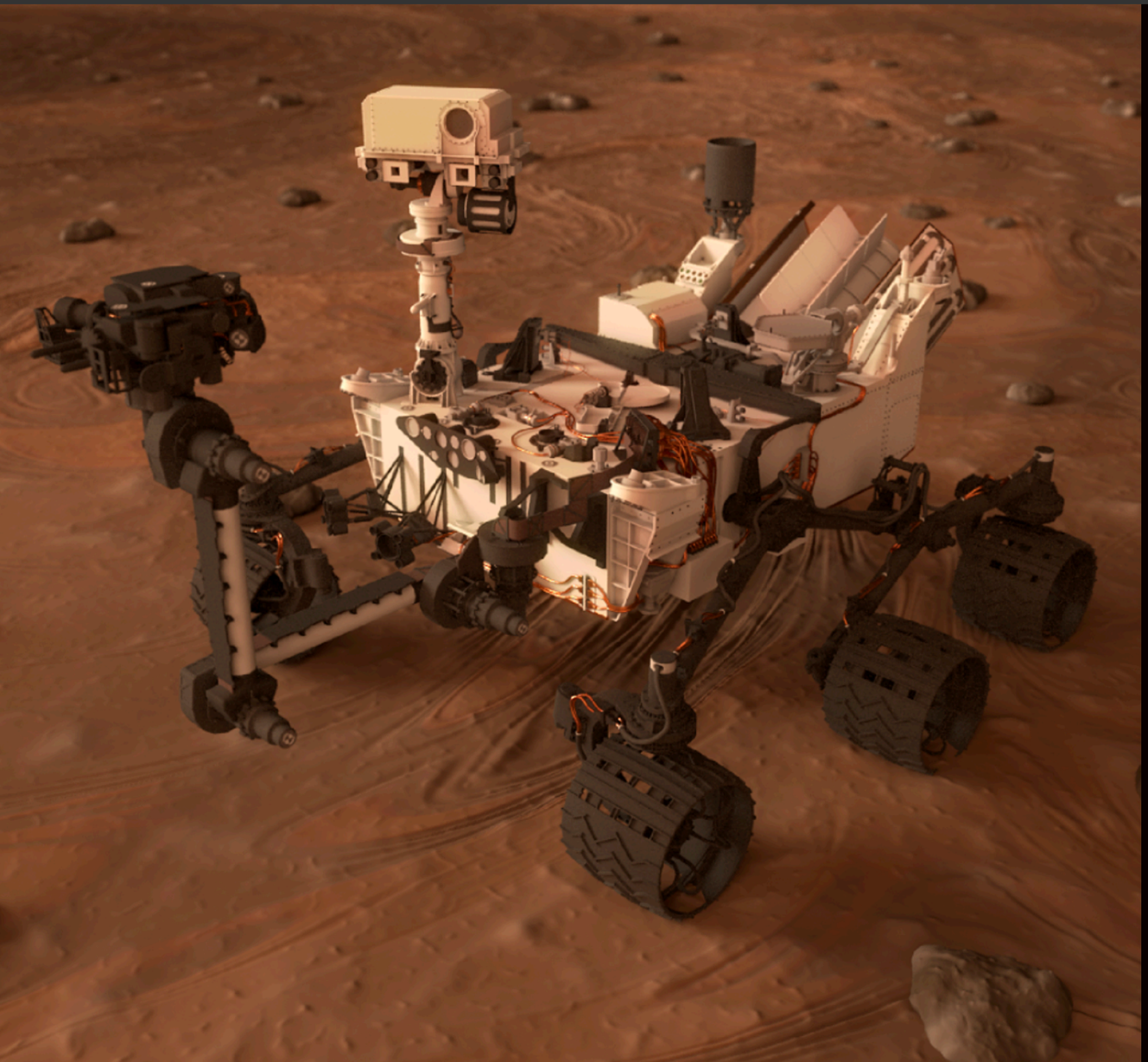


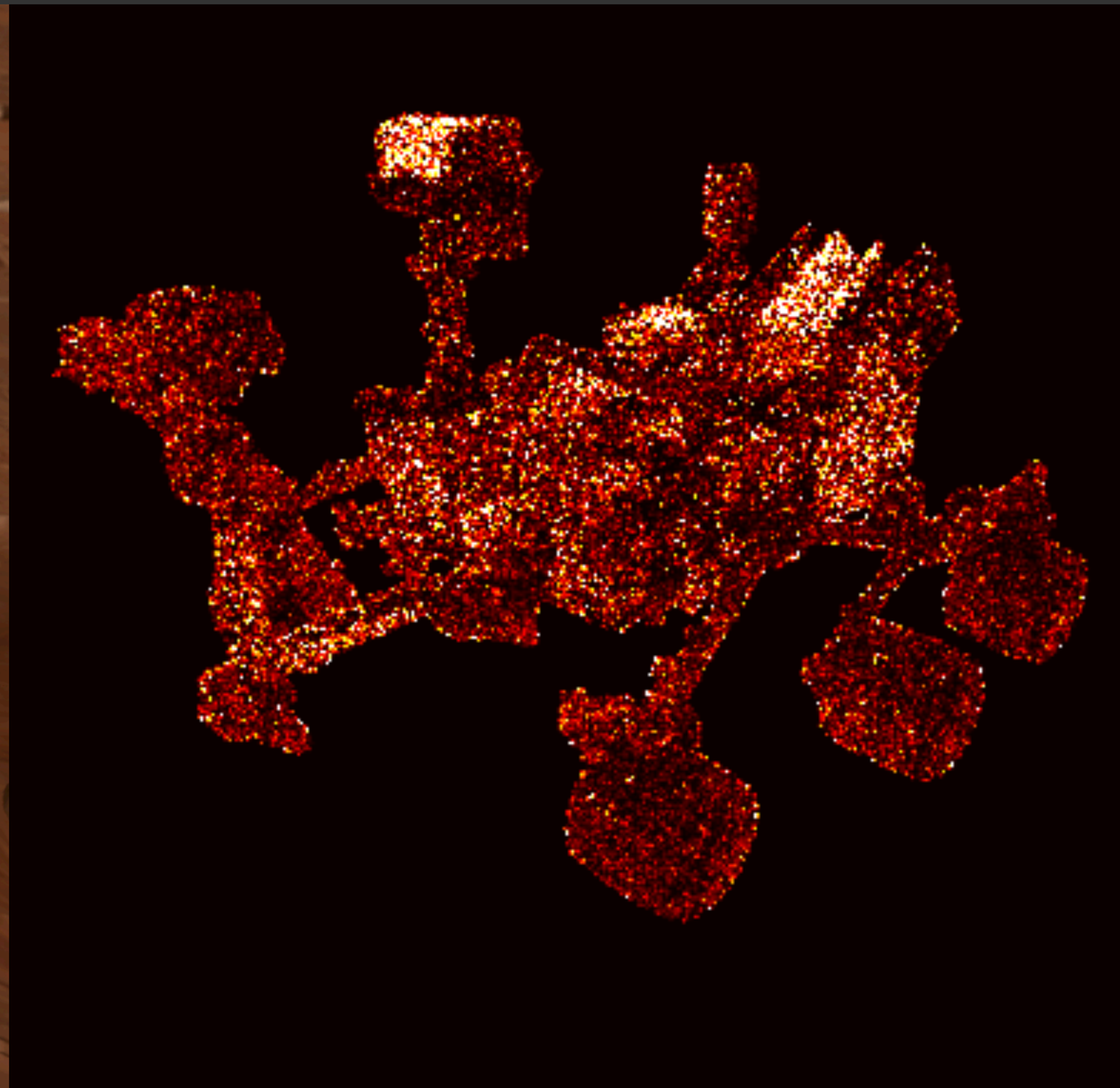
HARD



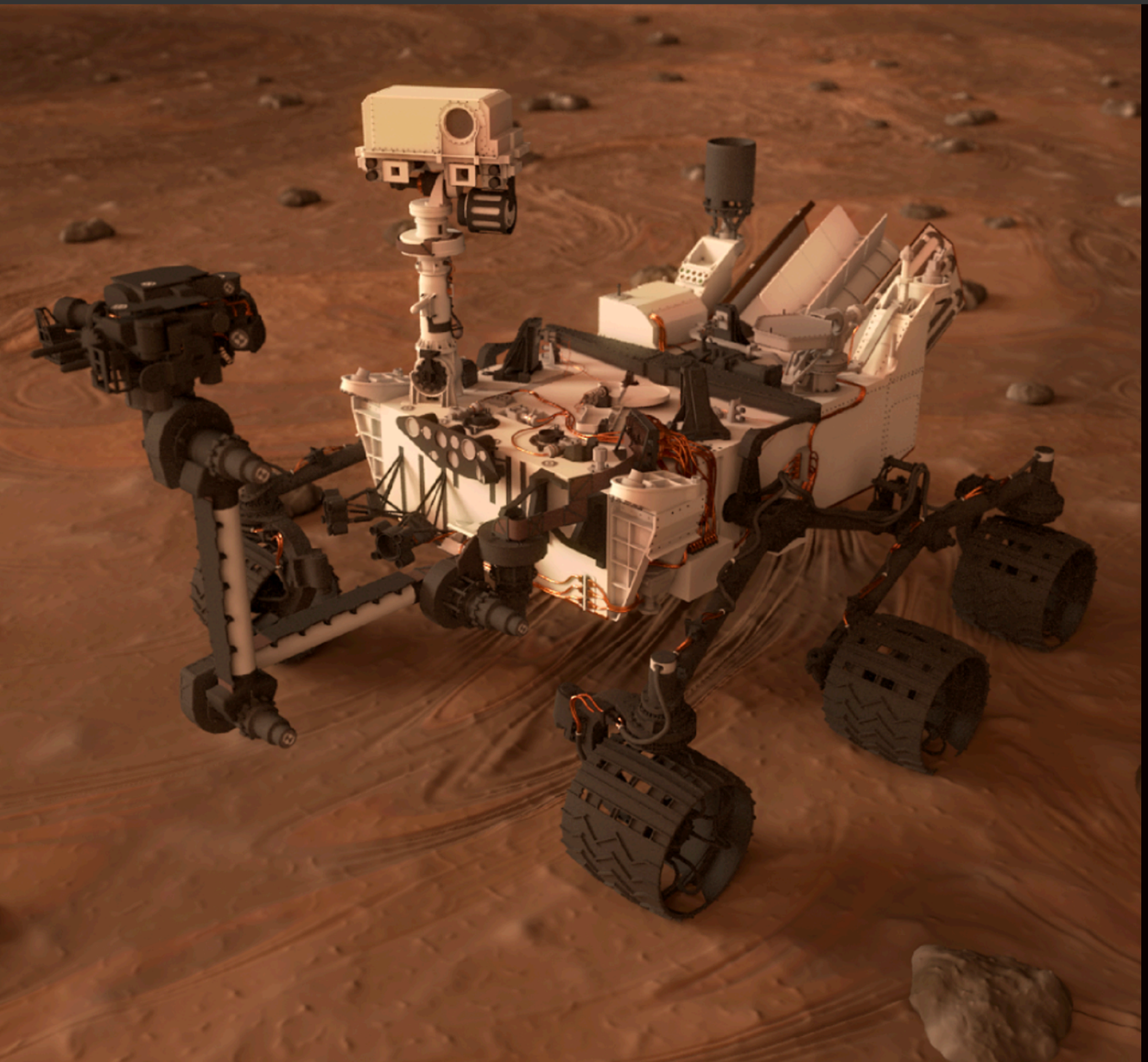
EASY

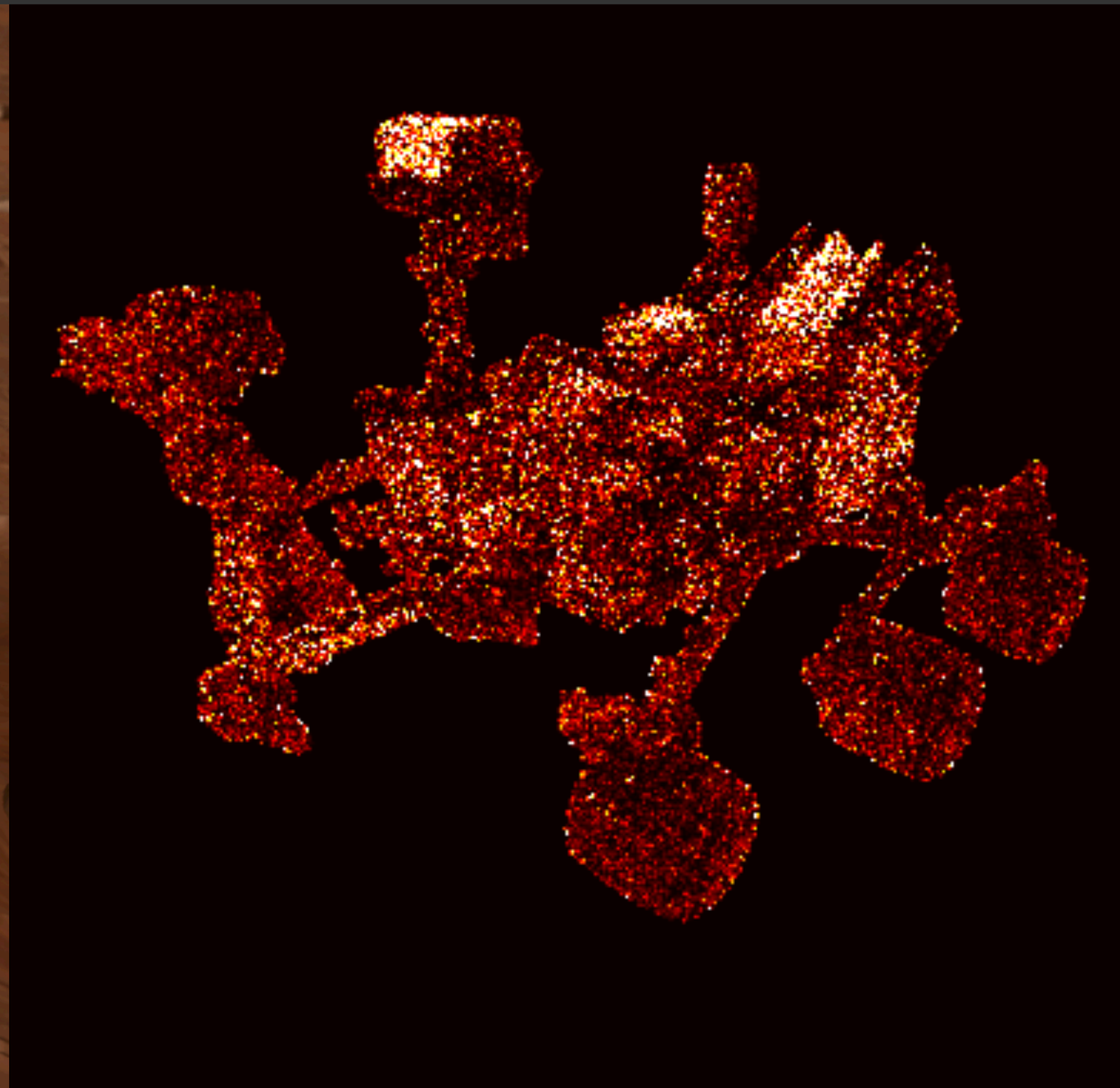
Thermal analysis of Curiosity Mars rover



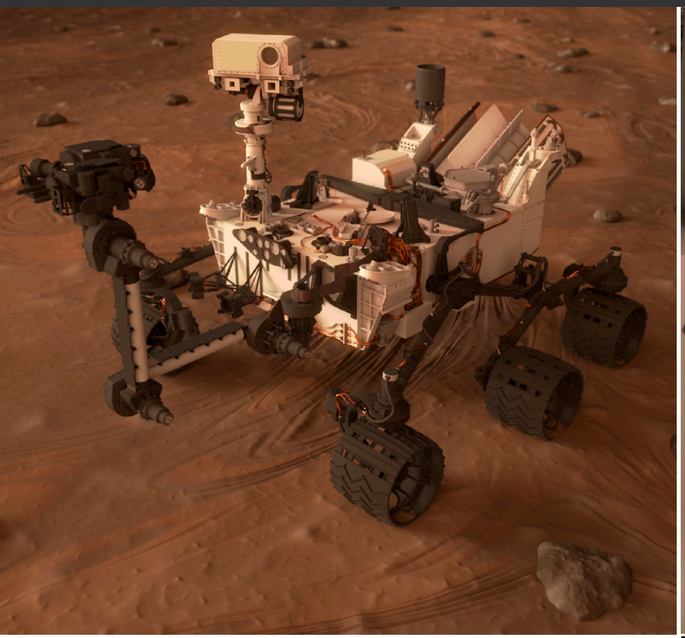
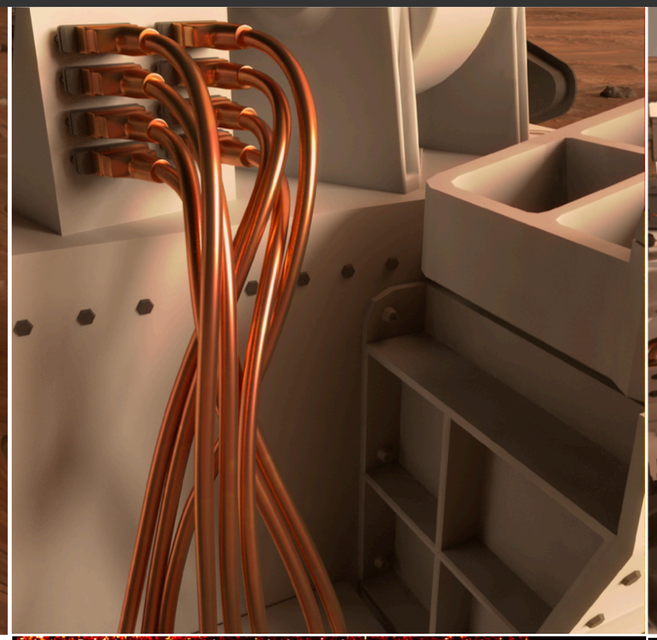
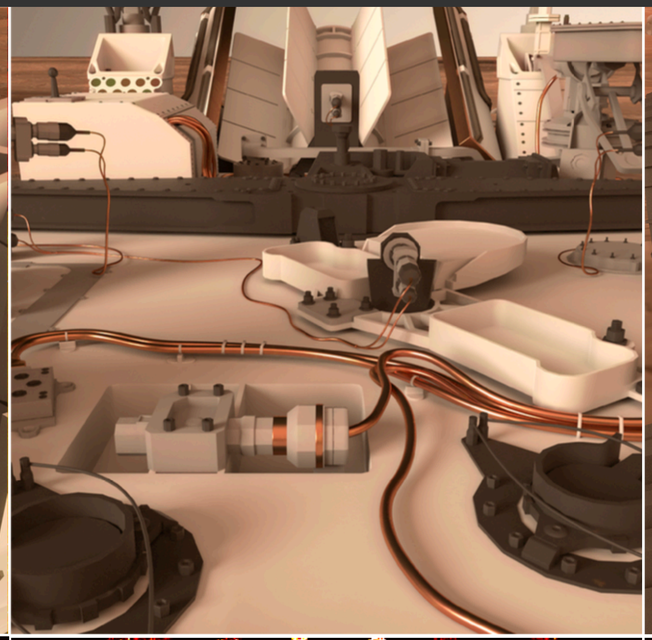
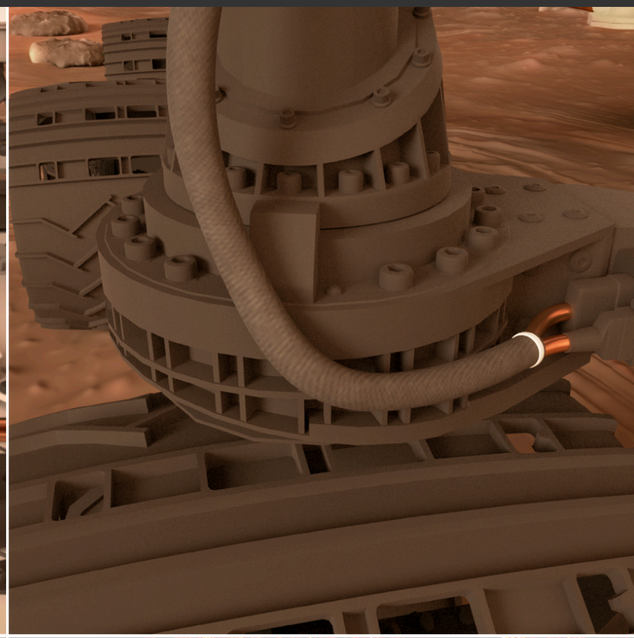


Thermal analysis of Curiosity Mars rover

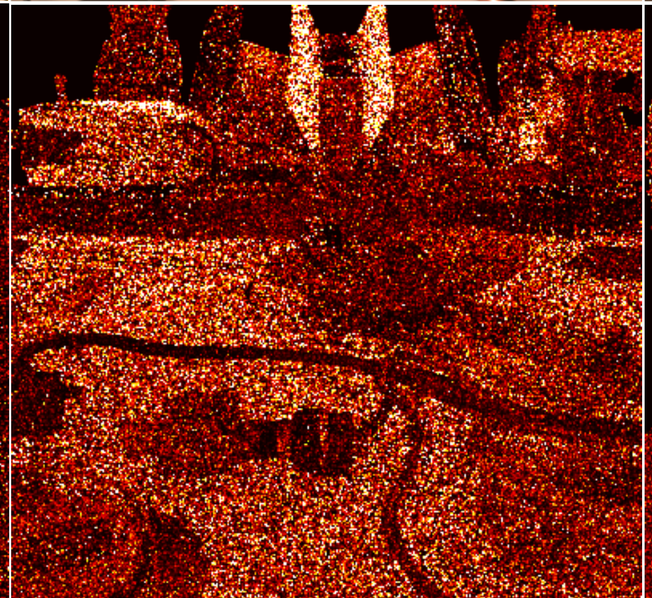
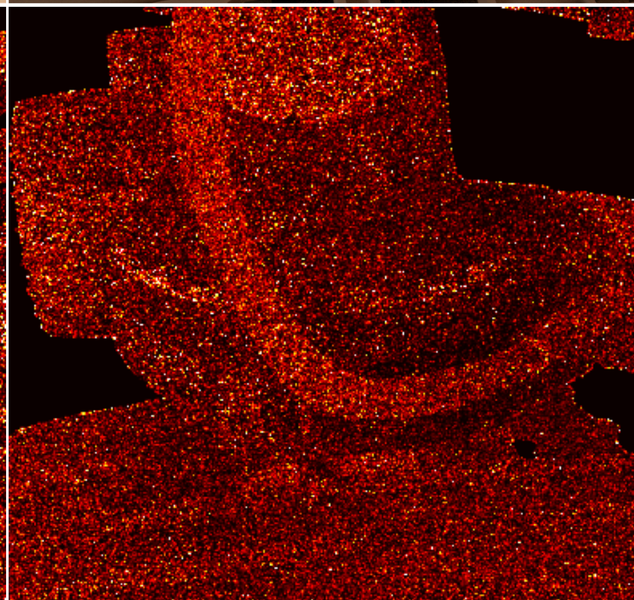




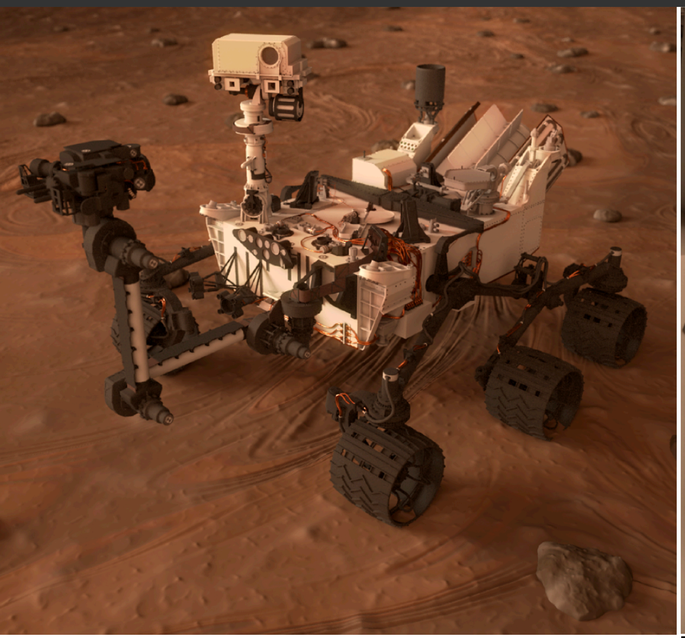
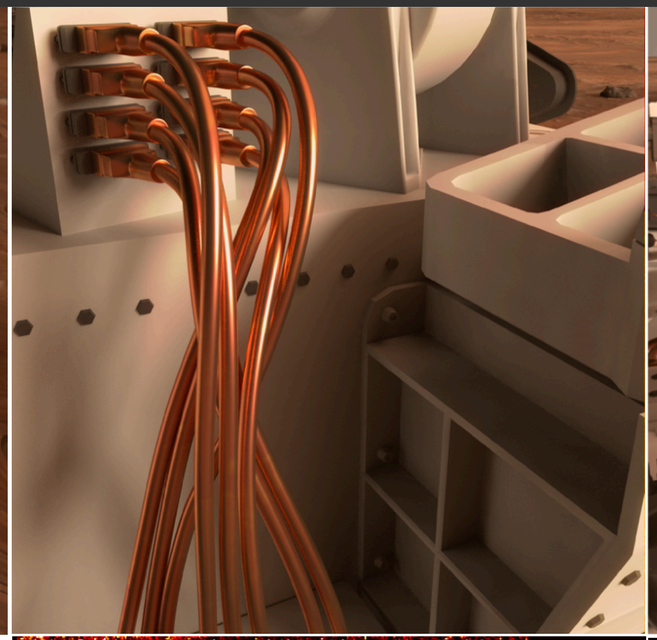
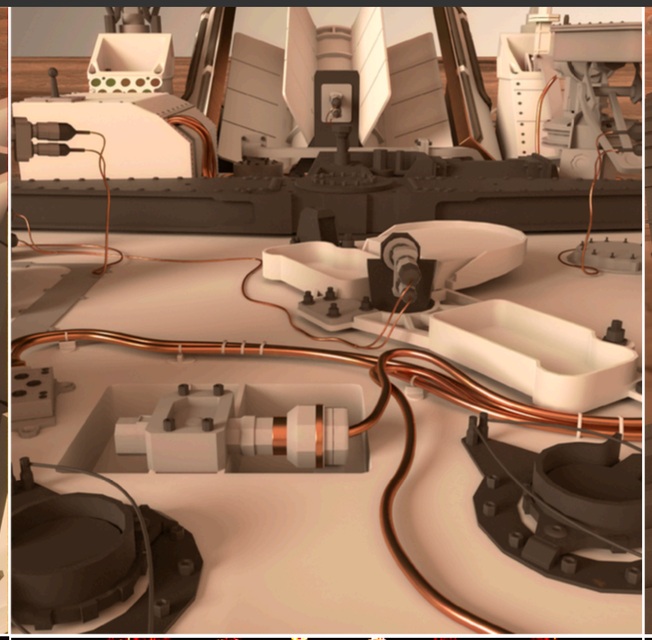
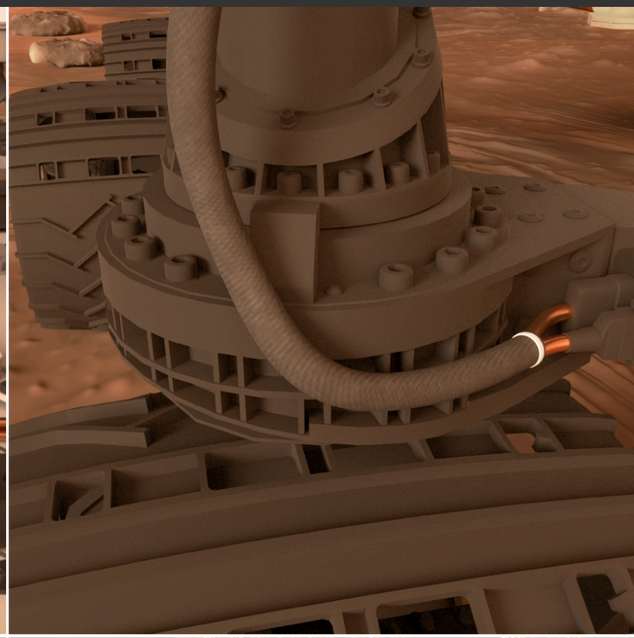
Simulate only what you see!

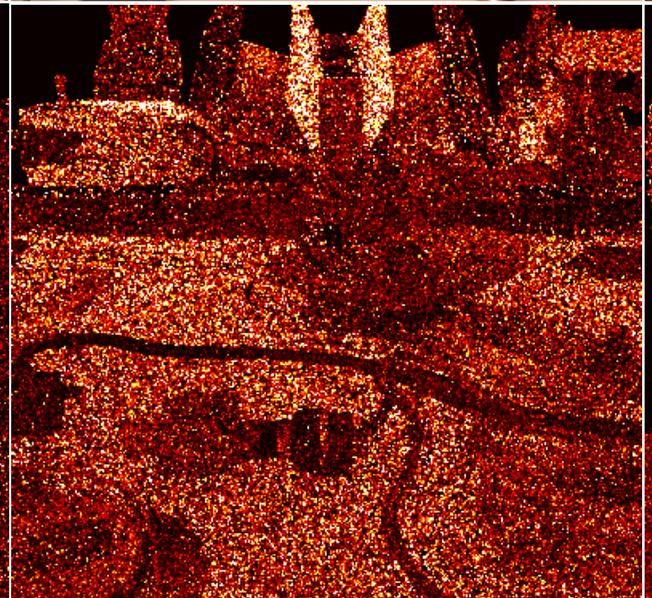
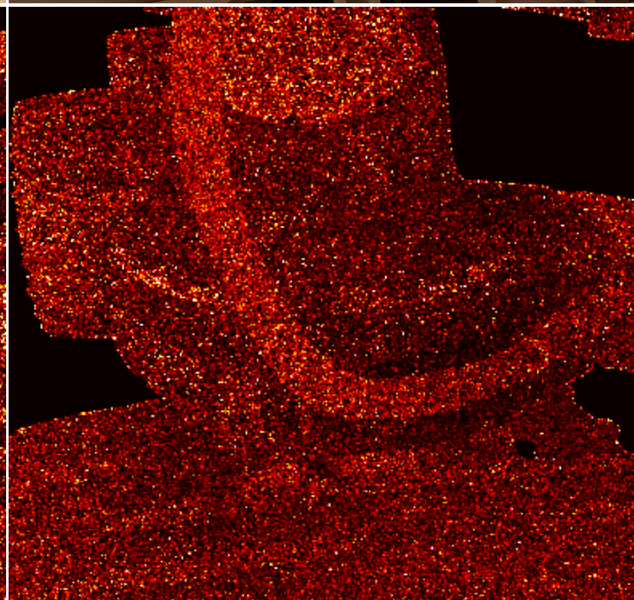
Analyze <u>locally</u>
in region
of interest!

Simulate only what you see!

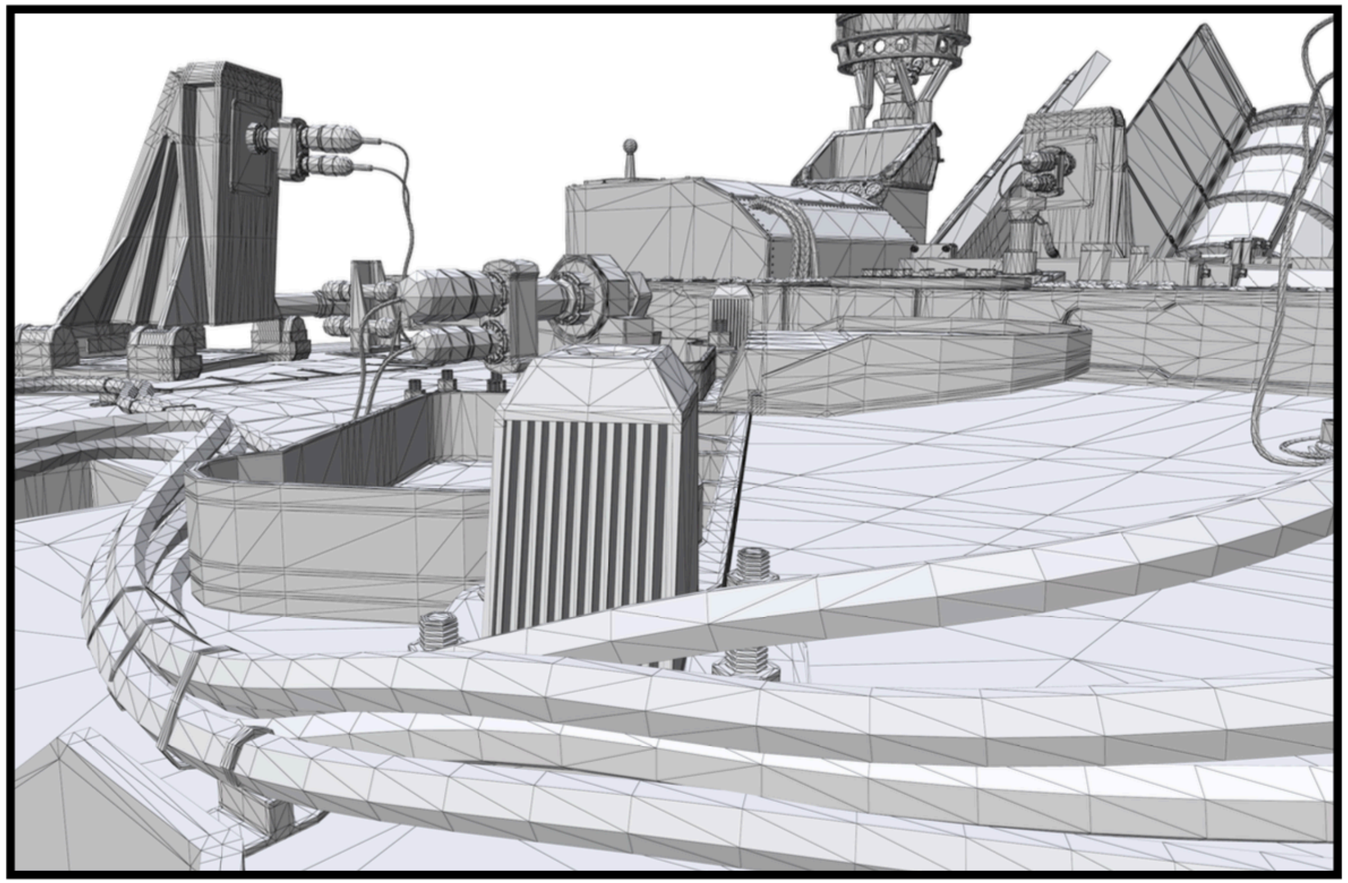





Analyze <u>locally</u>
in region
of interest!

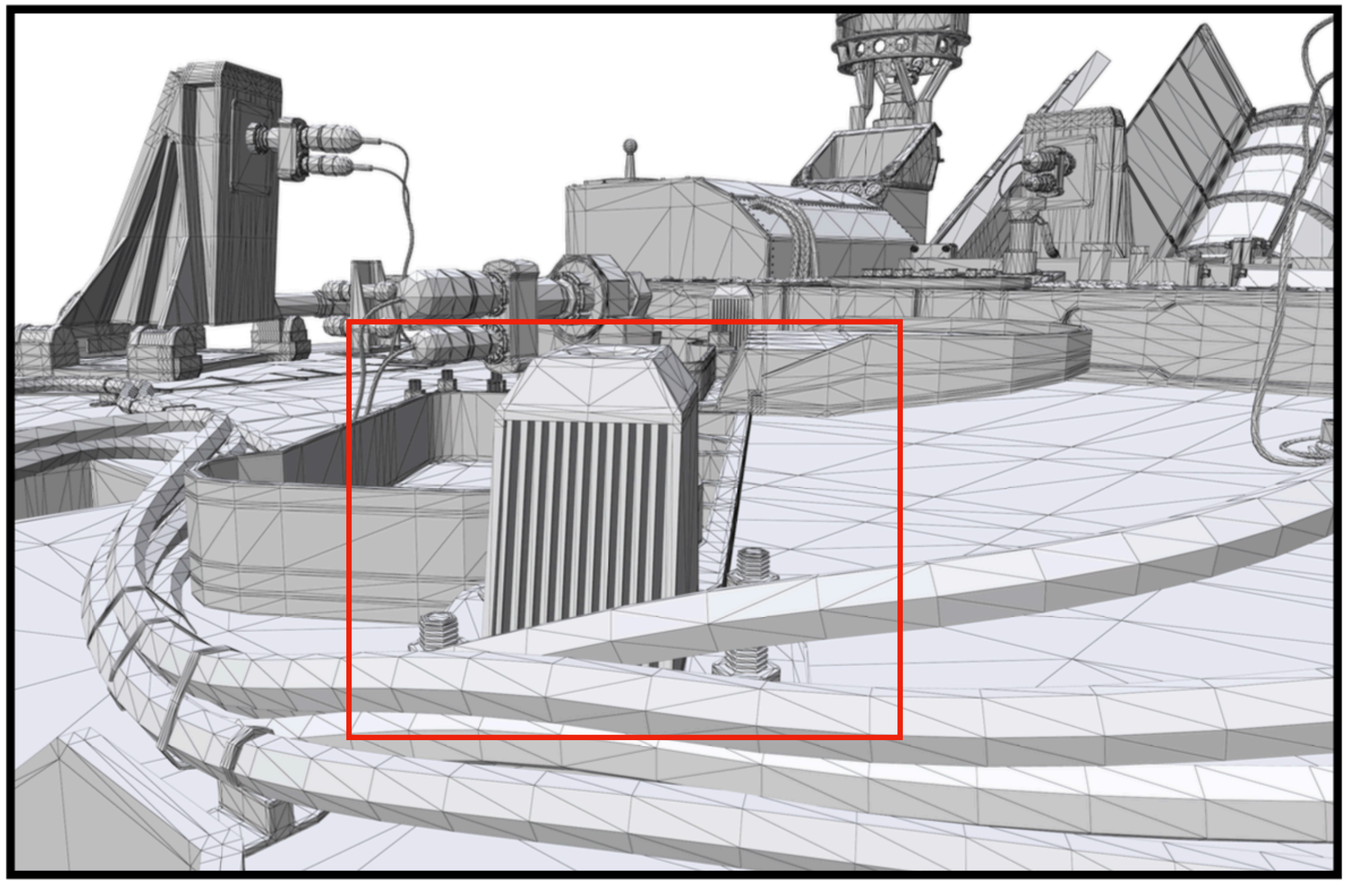
Thermal analysis is traditionally difficult in design phase

input boundary mesh

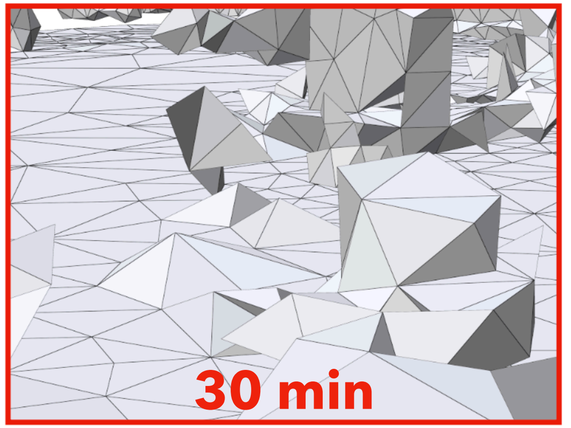


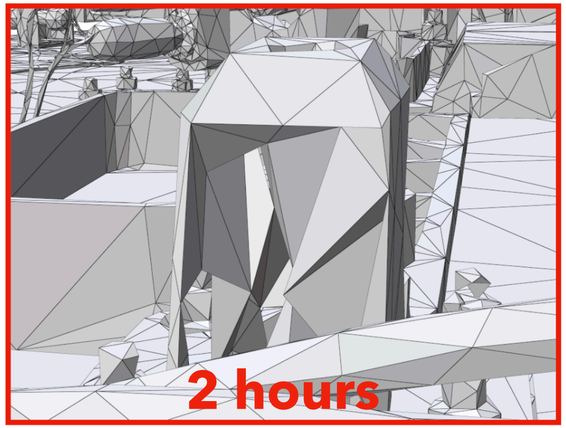
Thermal analysis is traditionally difficult in design phase

input boundary mesh



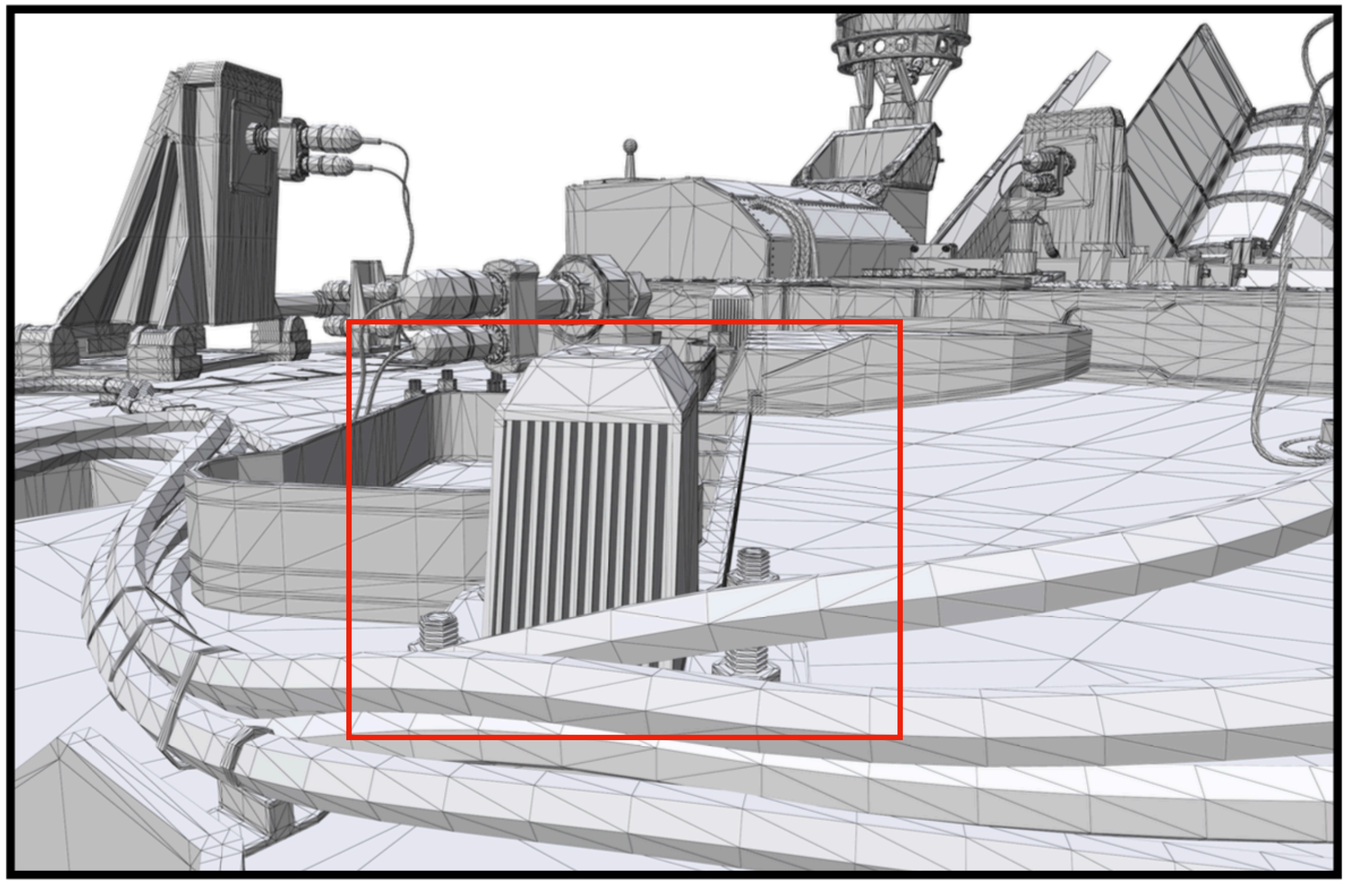
boundary of tetrahedral mesh



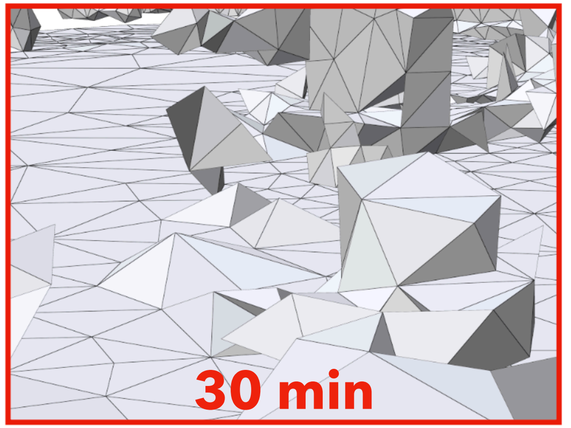


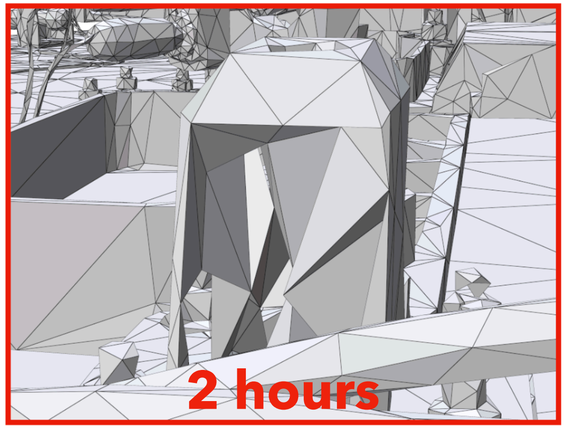
Thermal analysis is traditionally difficult in design phase

input boundary mesh



boundary of tetrahedral mesh

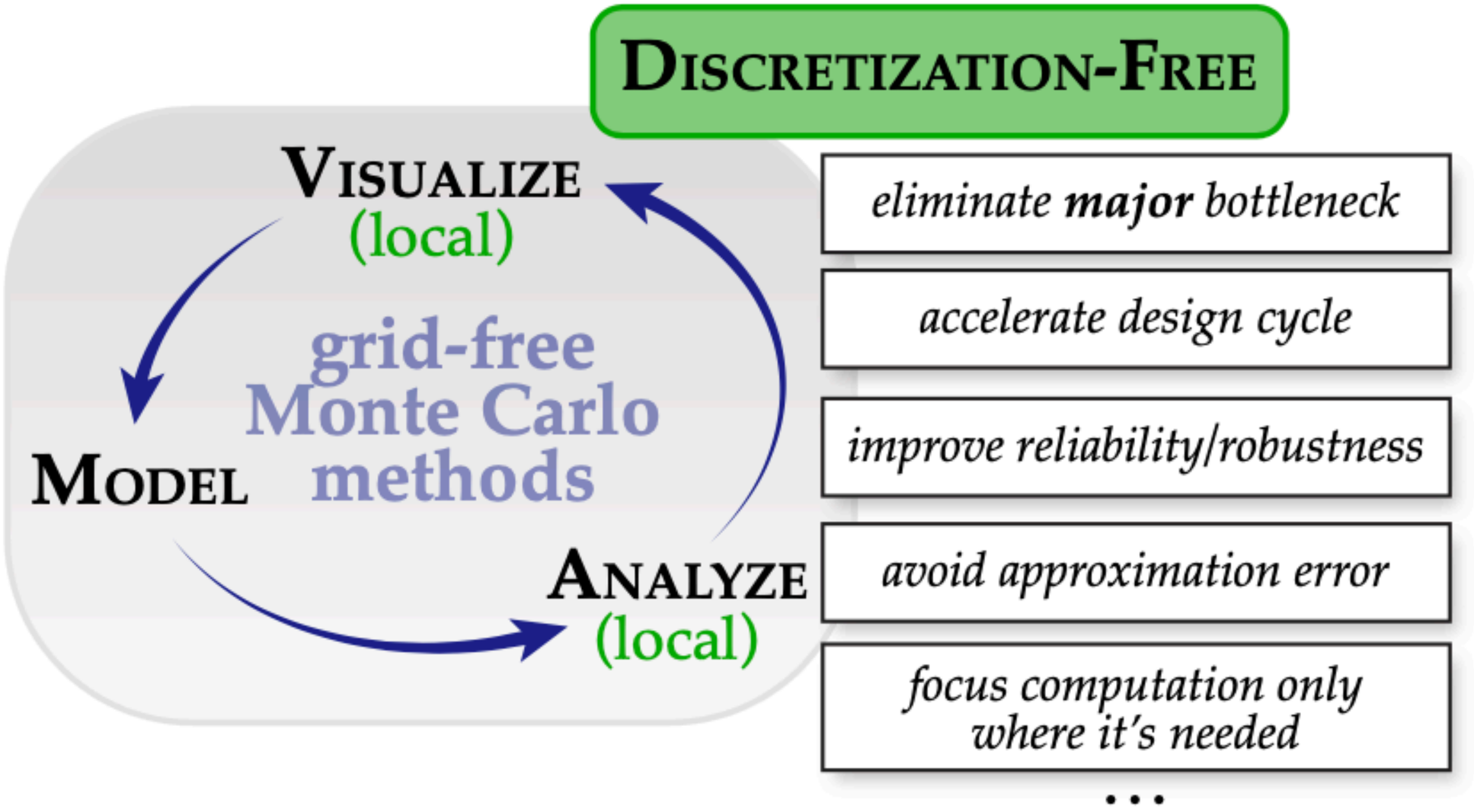




Out of memory

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)

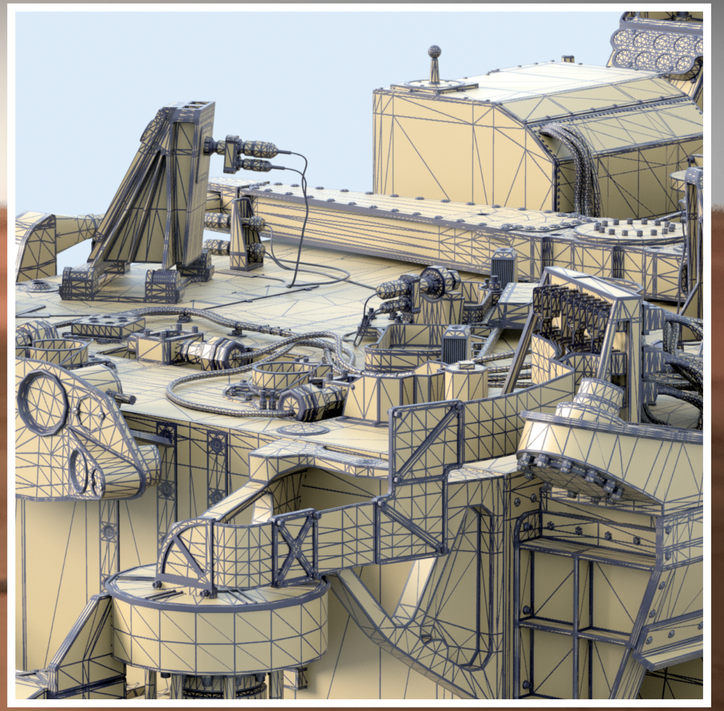
May never be able to solve certain PDEs w Monte Carlo…

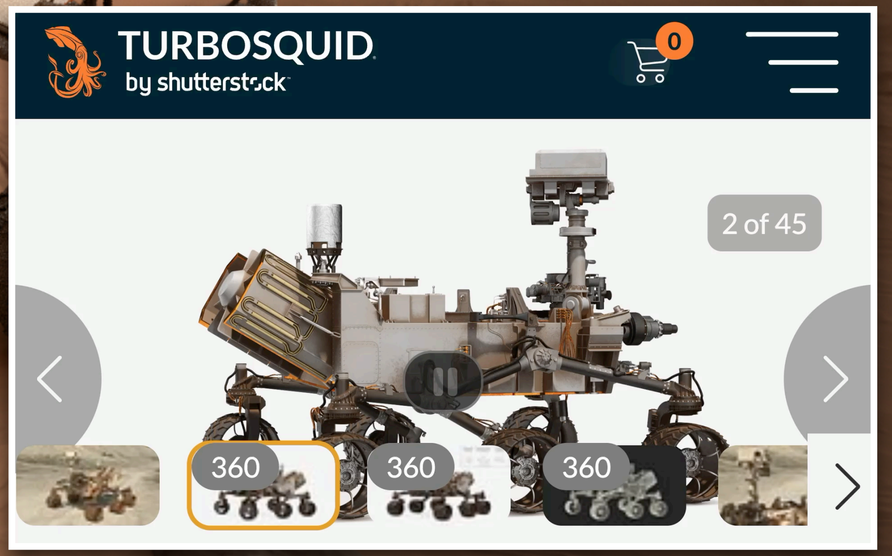
Examples from:

Han et al, “A Hybrid Material Point Method for Frictional Contact with Diverse Materials” (2019)

Zhu et al, “Codimensional Non-Newtonian Fluids” (2015)

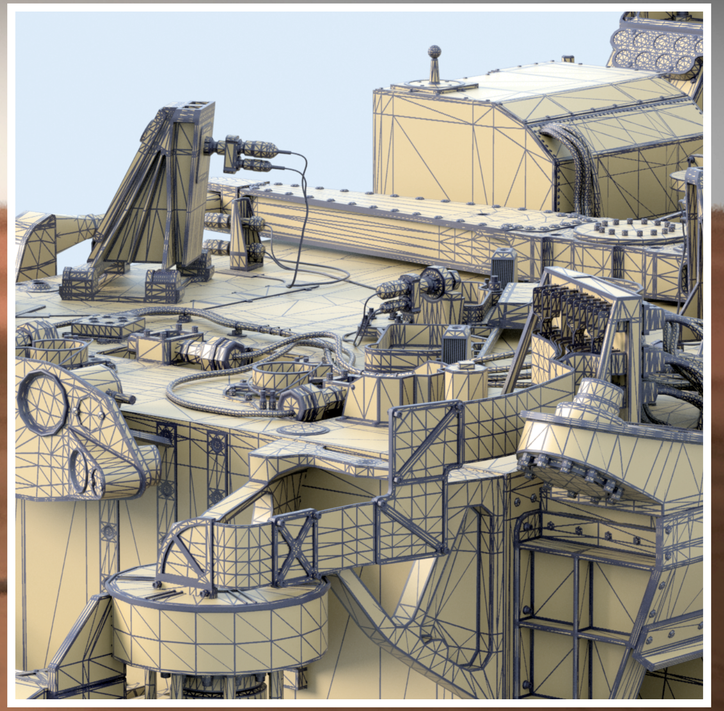
a rendering of NASA's *Curiosity* Mars Rover

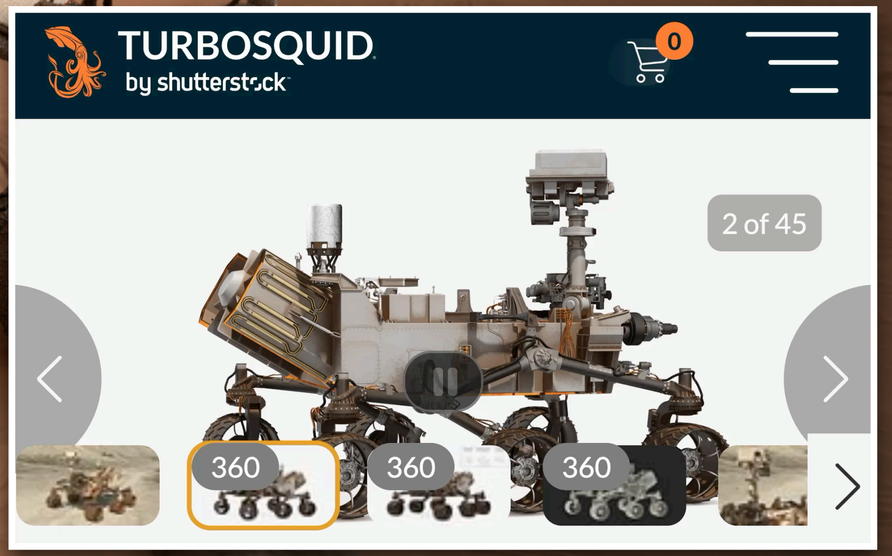




Rendering "just works," and gives immediate feedback, no matter what you throw at it.

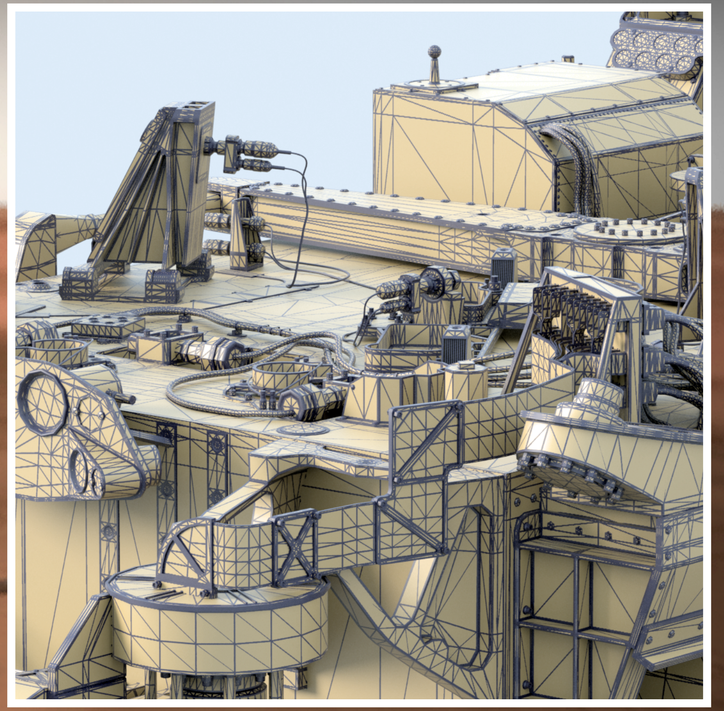
a rendering of NASA's *Curiosity* Mars Rover

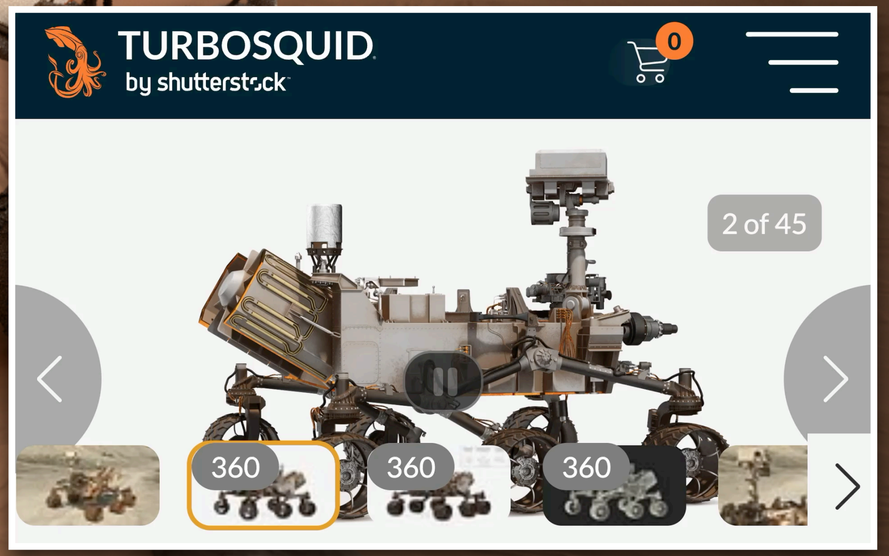




Rendering "just works," and gives immediate feedback, no matter what you throw at it.

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?

Overview

Overview

Parts 1: WoS for Laplace eq

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

Overview

Parts 1: WoS for Laplace eq

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

Parts 2: WoS for Poisson eq

– **key concept:** importance sampling

Overview

Parts 1: WoS for Laplace eq

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

Parts 2: WoS for Poisson eq

– **key concept:** importance sampling

Part 3: Variance reduction

PART 1:
WoS for Laplace Equation

Solving problems with integration

Solving problems with integration

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

Solving problems with integration

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

write down problem

Find u such that

$$\Delta u = f$$

Solving problems with integration

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

write down problem

Find u such that

$$\Delta u = f$$

rewrite as integral

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

Solving problems with integration

- Many problems can be reframed as integration problems: 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

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

Solving problems with integration

- Many problems can be reframed as integration problems: 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

Roadmap for solving Laplace PDE

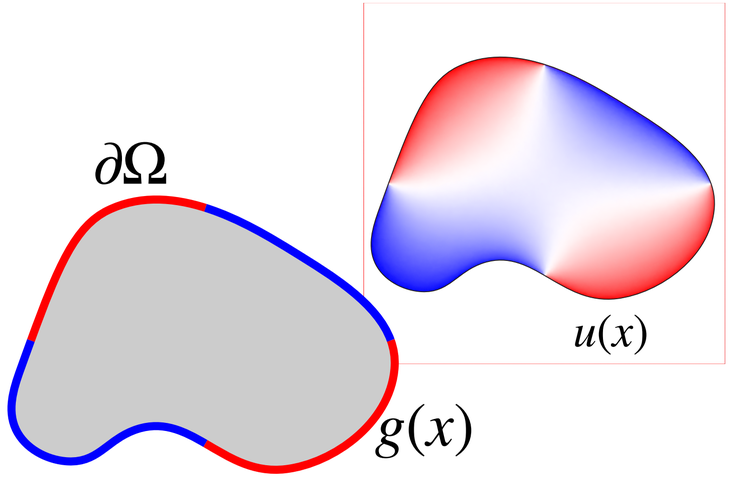
Roadmap for solving Laplace PDE

differential form

Laplace PDE

Find u such that

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



Roadmap for solving Laplace PDE

differential form

Laplace PDE

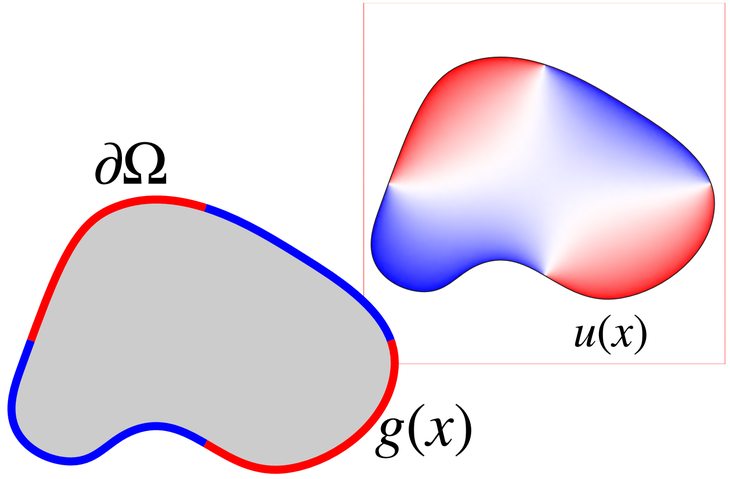
Find u such that

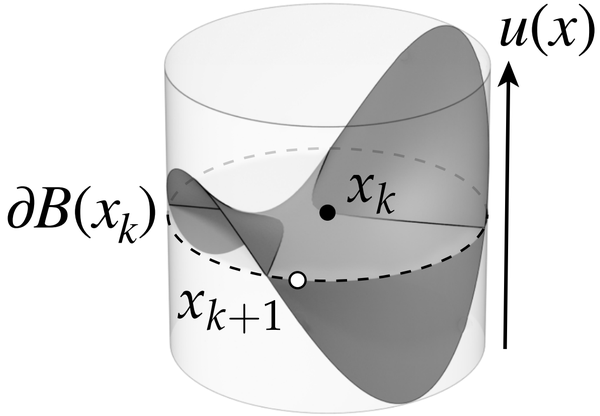
$$\Delta u = 0 \quad \text{on } \Omega$$
$$u = g \quad \text{on } \partial\Omega$$

integral representation

mean value integral

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





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

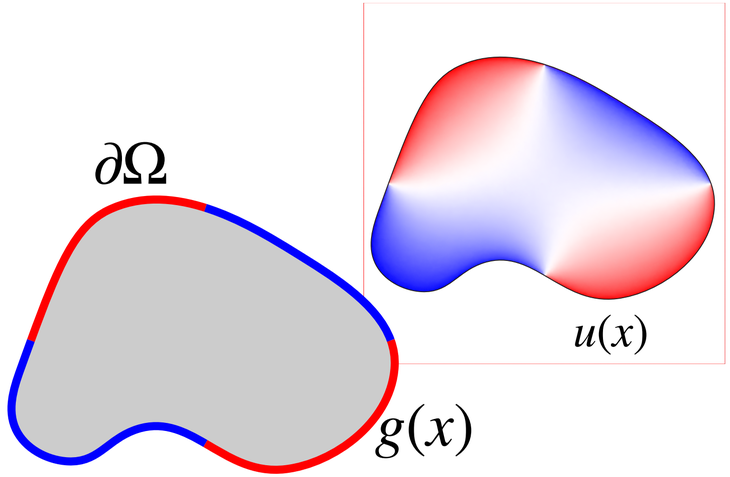
mean value integral

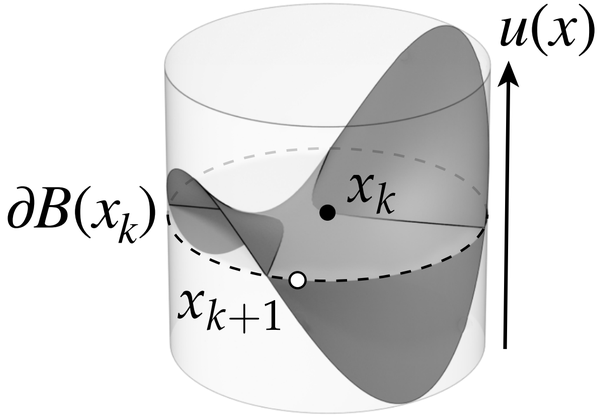
$$u(x_k) = \frac{1}{|\partial B(x_k)|} \int_{\partial B(x_k)} u(y)\, dy$$

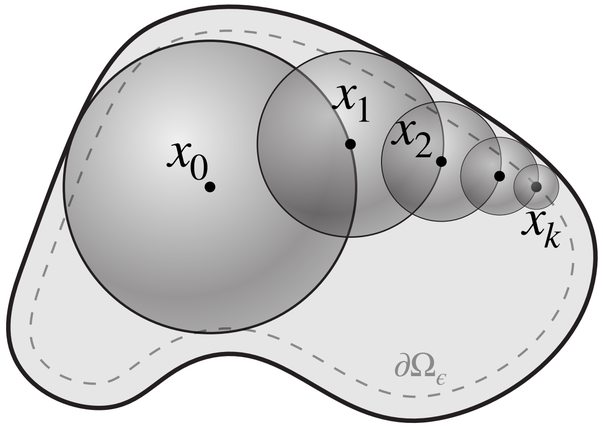
Monte Carlo estimator

Walk on Spheres

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





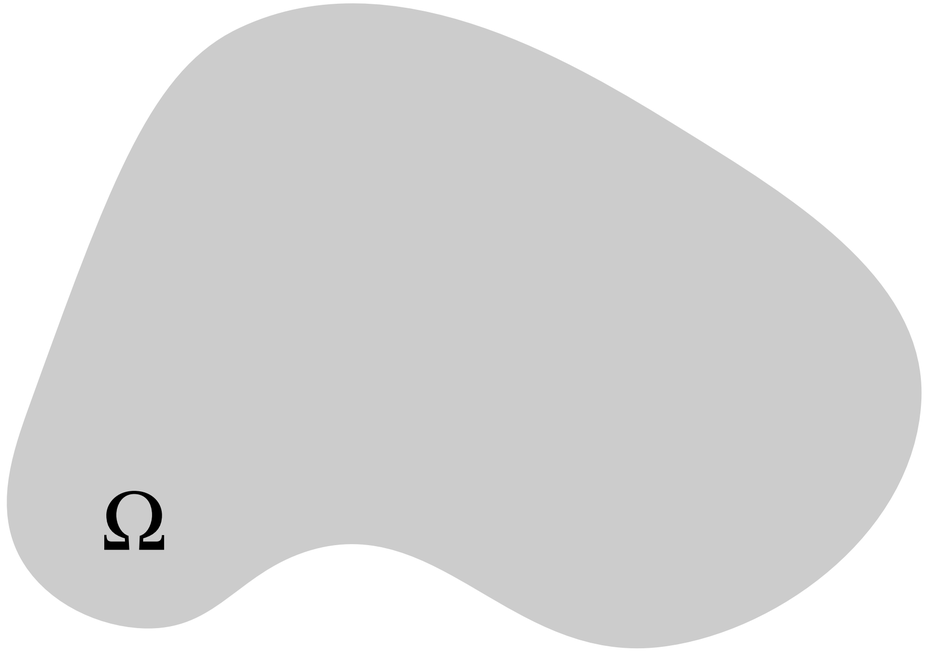


Laplace PDE with Dirichlet boundary condition

given: values on the boundary of a region Ω

find: smooth interpolation into interior

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

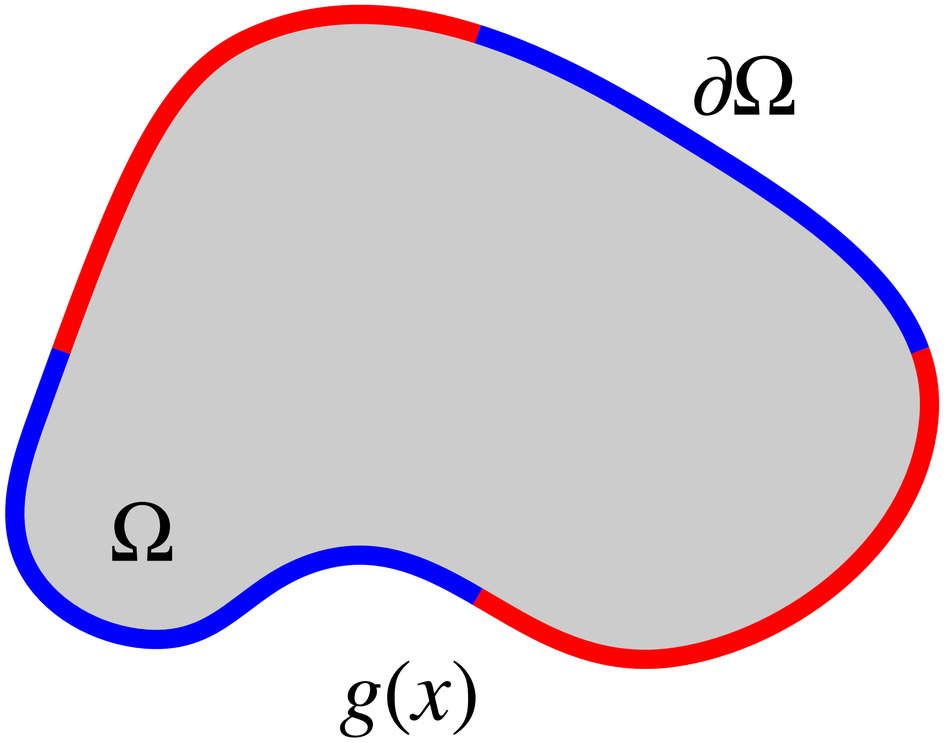


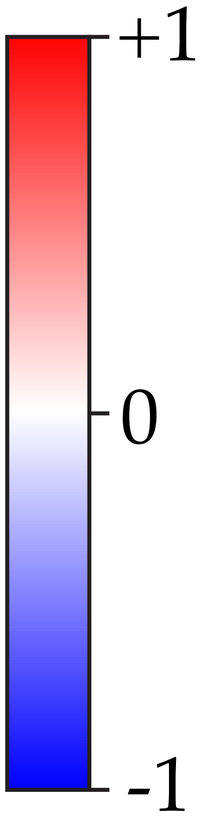
Laplace PDE with Dirichlet boundary condition

given: values on the boundary of a region Ω

find: smooth interpolation into interior

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



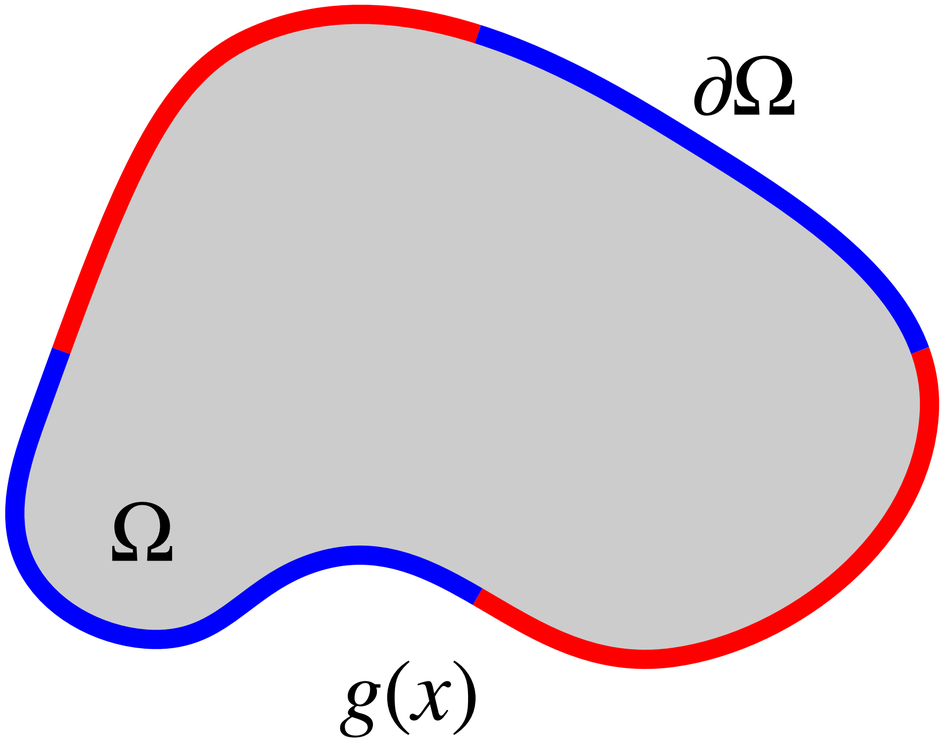


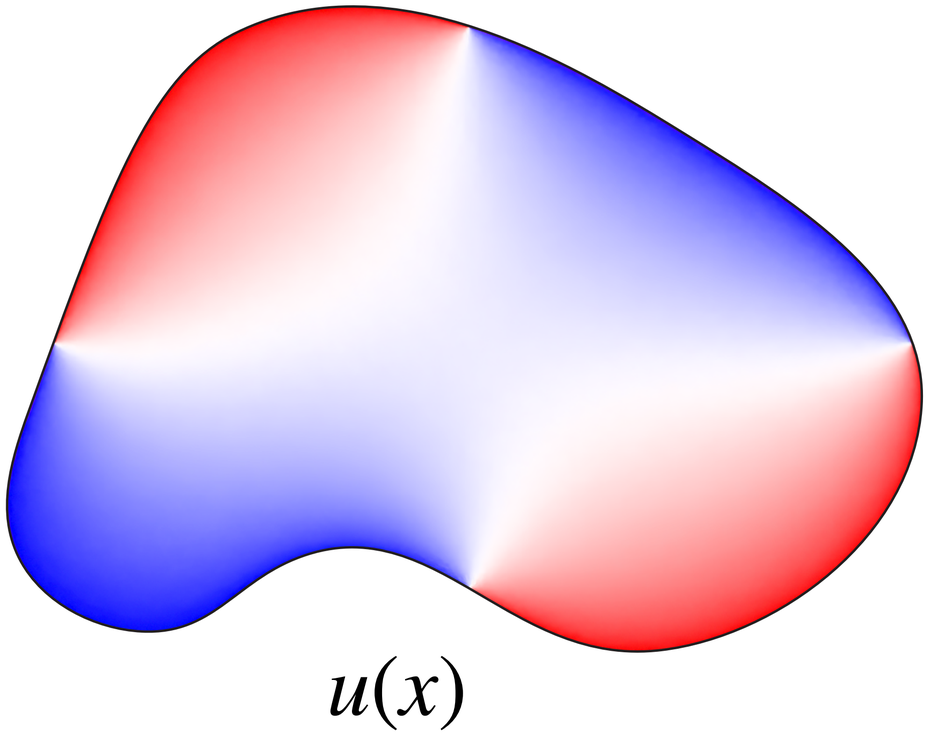
Laplace PDE with Dirichlet boundary condition

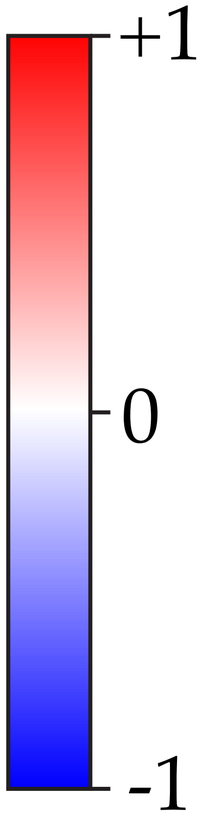
given: values on the boundary of a region Ω

find: smooth interpolation into interior

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





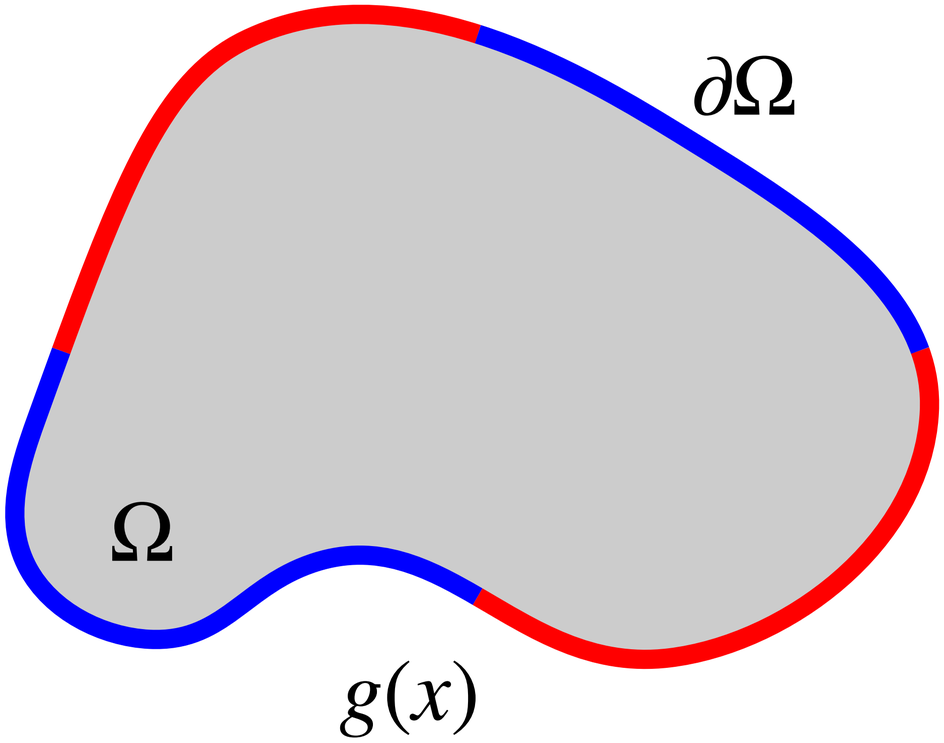


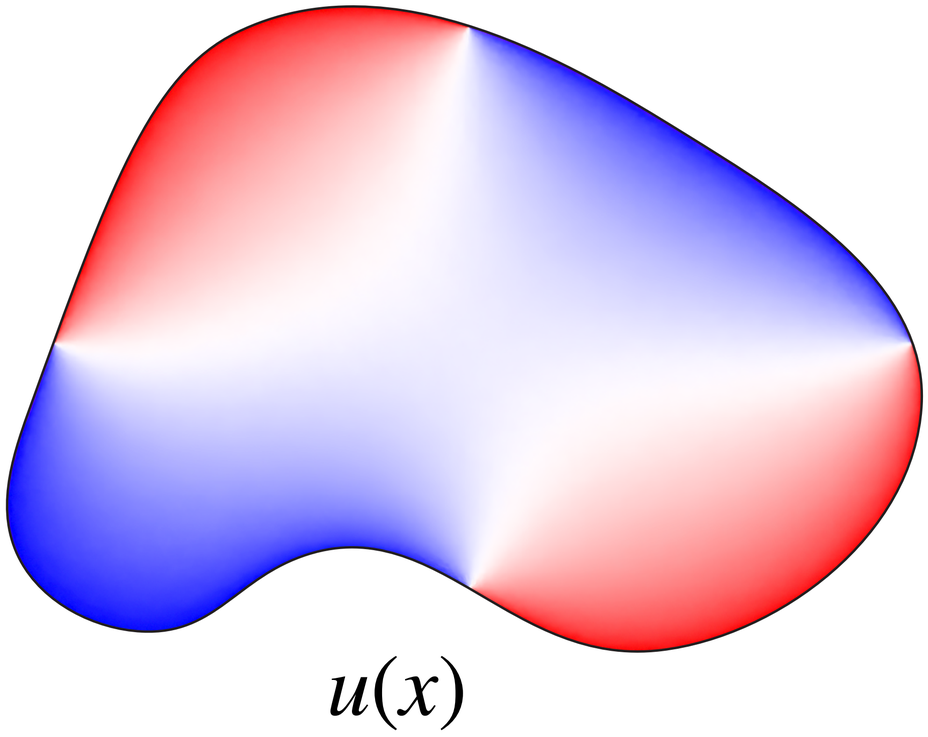
Laplace PDE with Dirichlet boundary condition

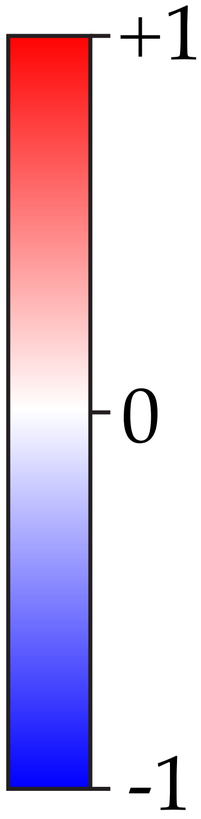
given: values on the boundary of a region Ω

find: smooth interpolation into interior

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





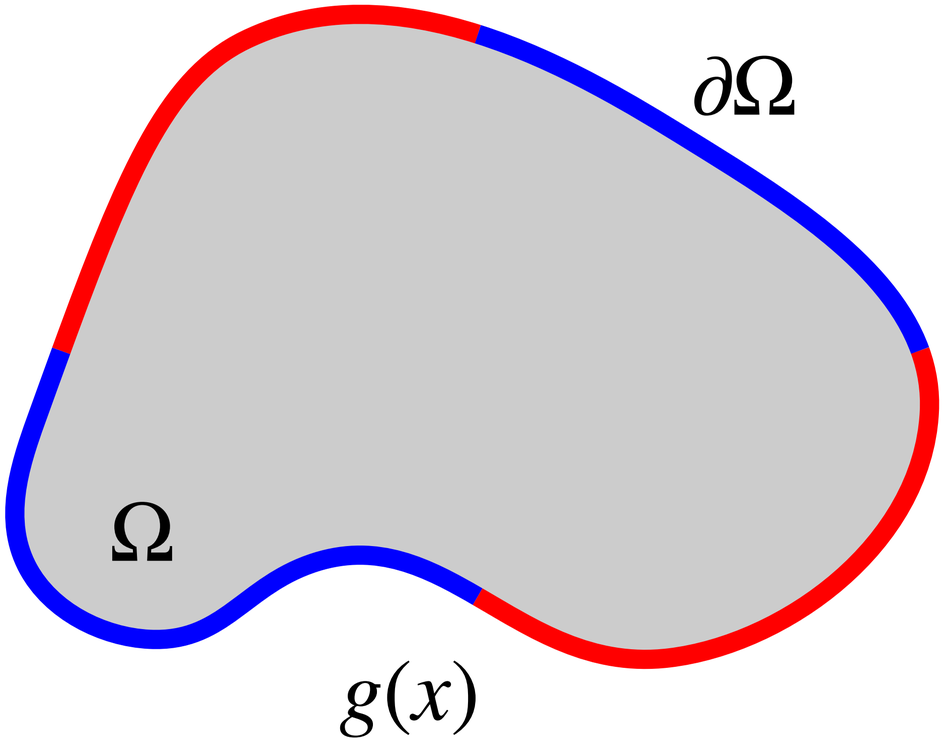


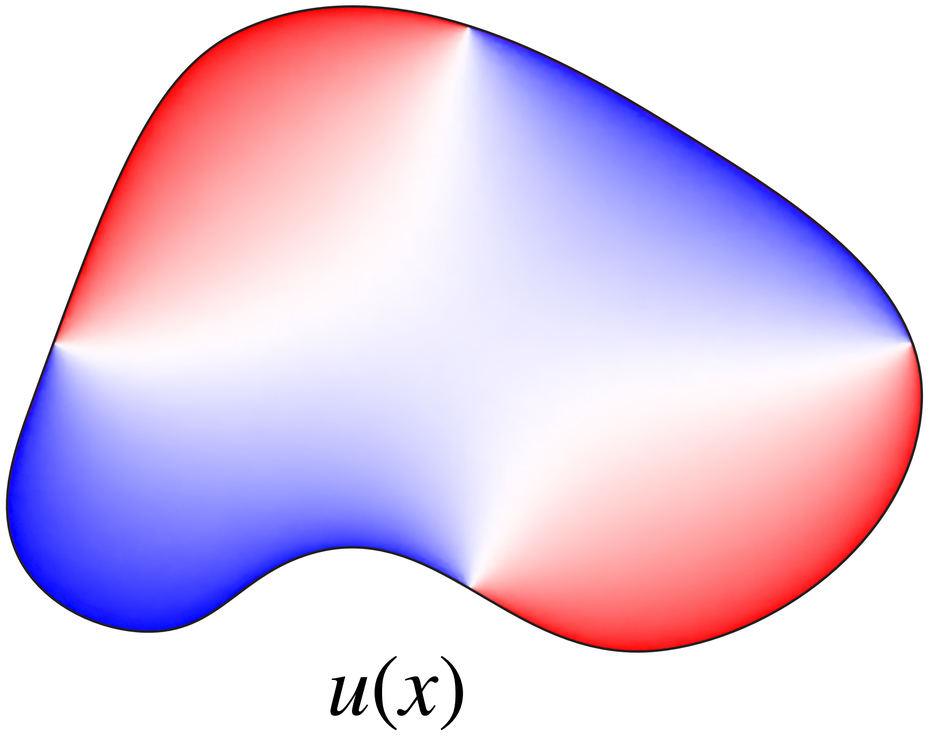
Laplace PDE with Dirichlet boundary condition

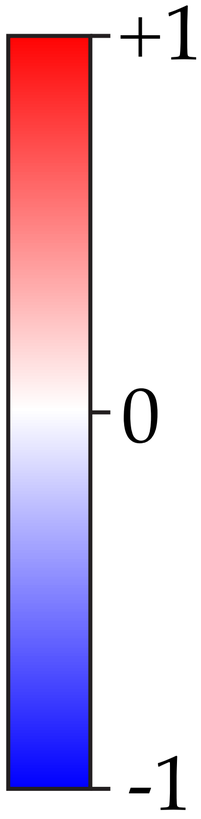
given: values on the boundary of a region Ω

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.

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

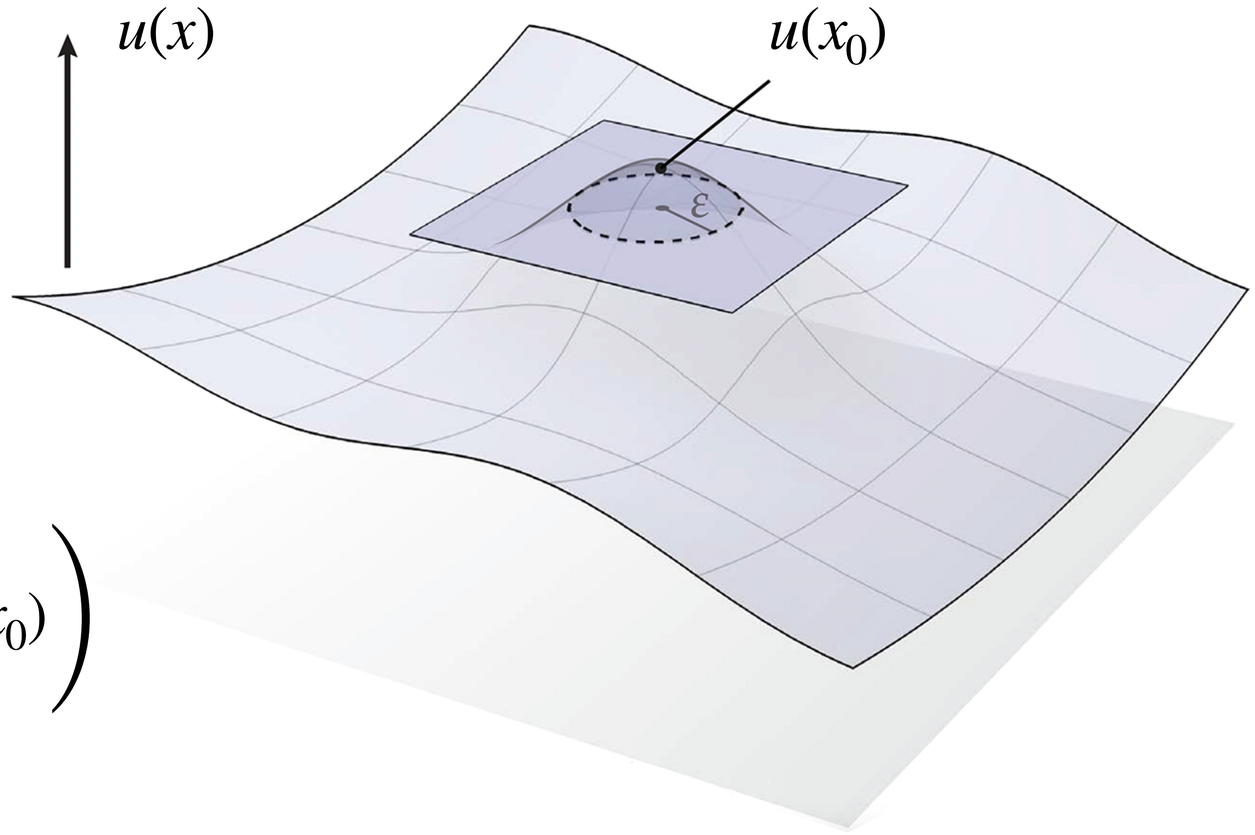
$$\Delta u(x, y) = \frac{\partial^2}{\partial x^2} u(x, y) + \frac{\partial^2}{\partial y^2} u(x, y)$$

differential operators.

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

Laplacian gives deviation from local average

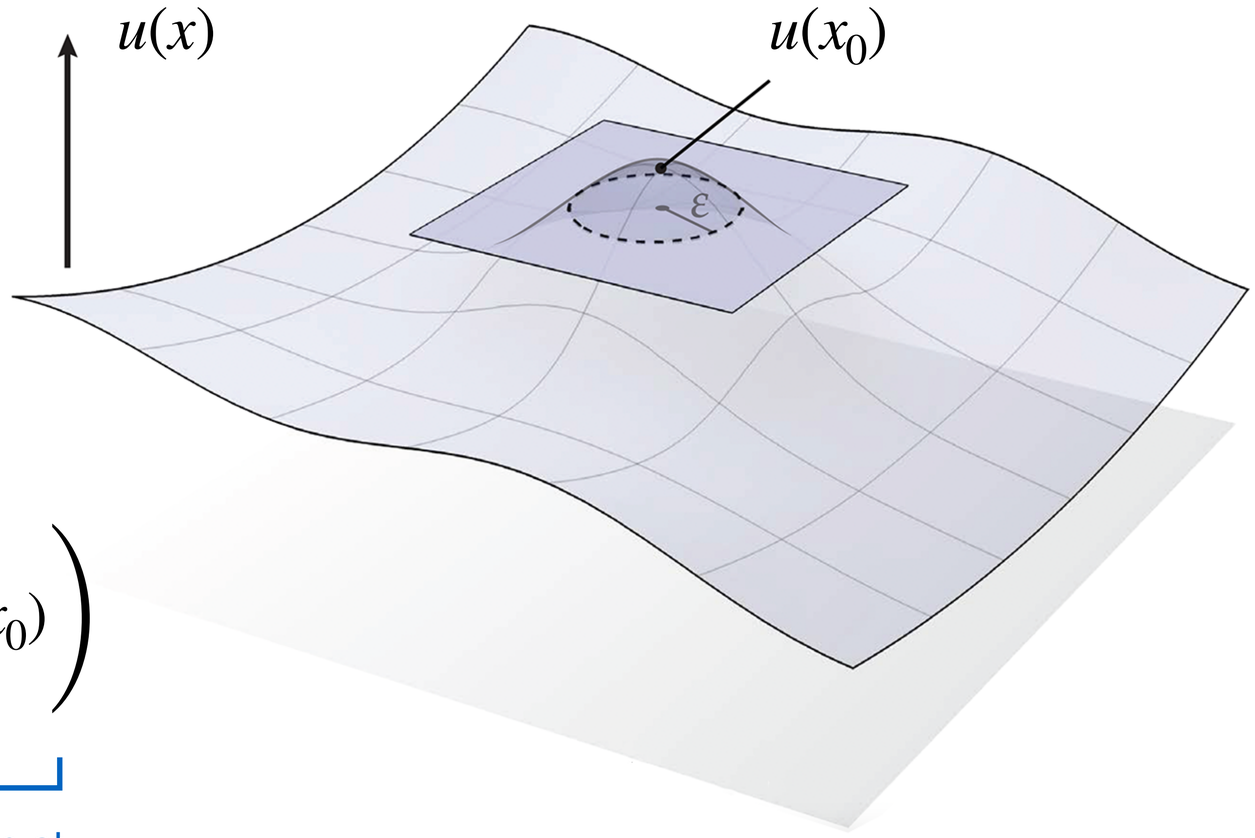
More intuitively, can think of the Laplacian of a function u as difference between value at a point x_0, and the average value over a small sphere (or ball) around x_0



$$\Delta u(x_0) \propto \lim_{\epsilon \to 0} \frac{1}{\epsilon^2} \left(\frac{1}{|S_\epsilon(x_0)|} \int_{S_\epsilon(x_0)} u(x)\, \mathrm{d}x - u(x_0) \right)$$

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

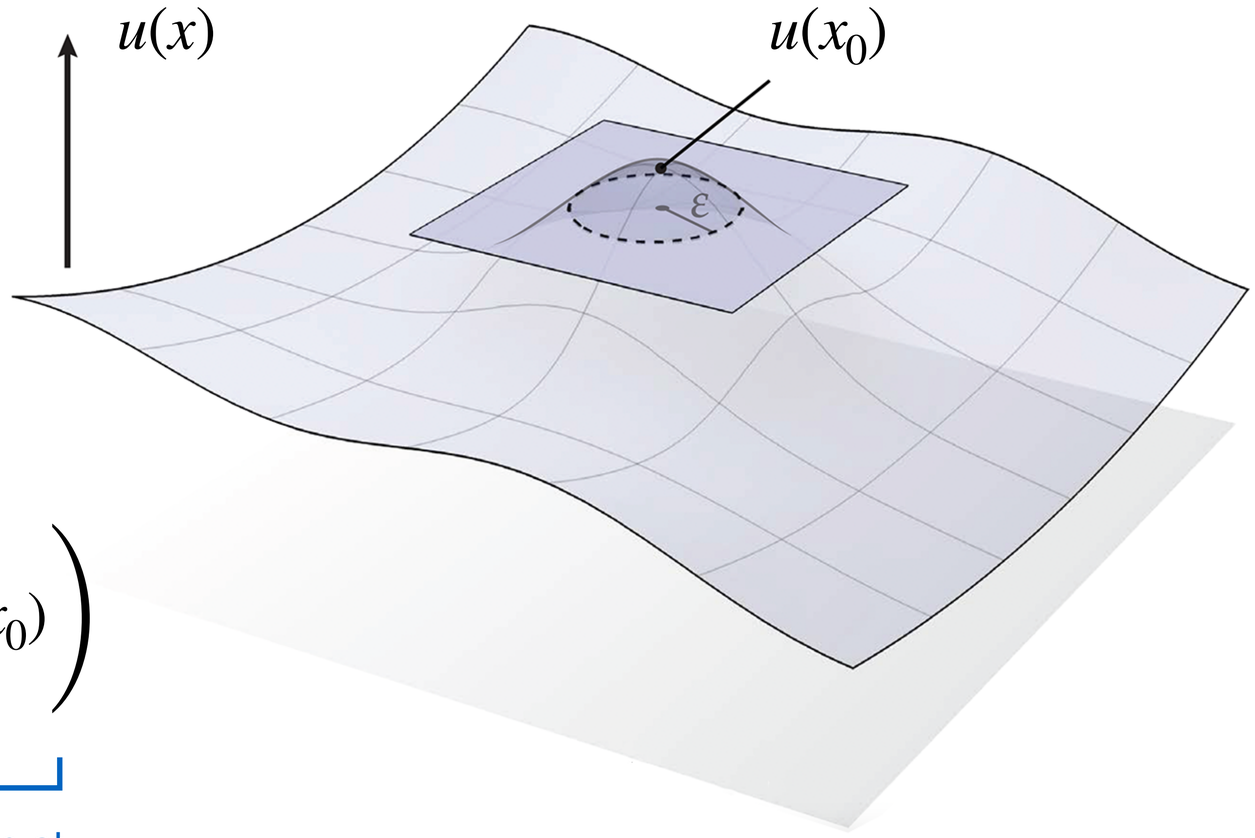


$$\Delta u(x_0) \propto \lim_{\epsilon \to 0} \frac{1}{\epsilon^2} \left(\frac{1}{|S_\epsilon(x_0)|} \int_{S_\epsilon(x_0)} u(x)\, \mathrm{d}x - u(x_0) \right)$$

value at center

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



$$\Delta u(x_0) \propto \lim_{\epsilon \to 0} \frac{1}{\epsilon^2} \left(\frac{1}{|S_\epsilon(x_0)|} \int_{S_\epsilon(x_0)} u(x)\, \mathrm{d}x - u(x_0) \right)$$

sphere area

integral over sphere

value at center

Mean value property of harmonic functions

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 $\Delta u = 0$) ?

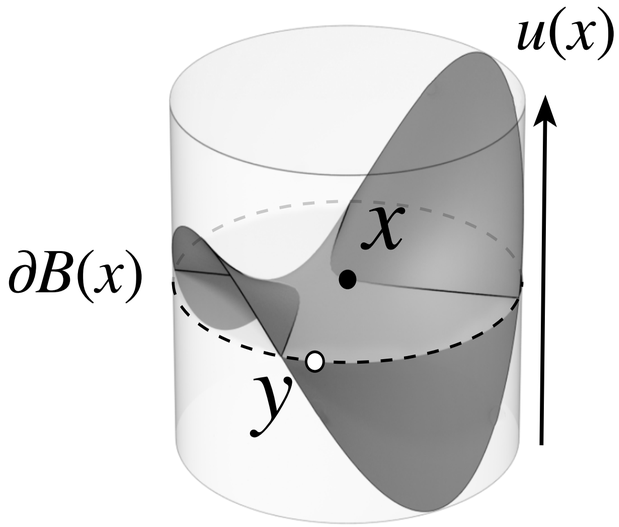
Mean value property of harmonic functions

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.

mean value property

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



Mean value property of harmonic functions

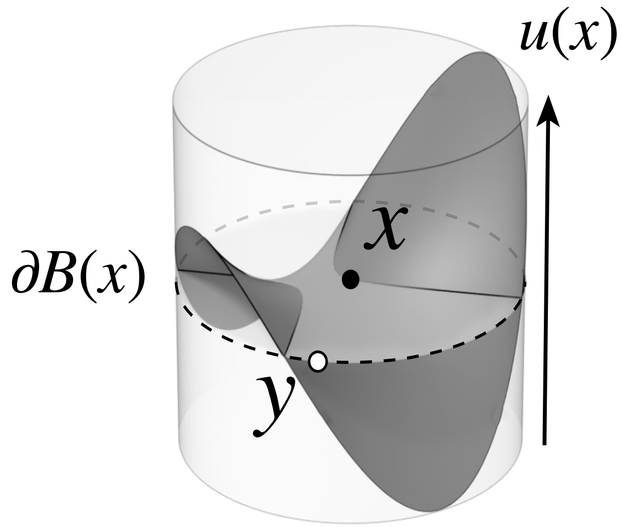
Q: Given this interpretation of the Laplacian, what can we say about the behavior of a ***harmonic function*** (i.e. a function u satisfying $\Delta u = 0$) ?

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

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

mean value property

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



Mean value property of harmonic functions

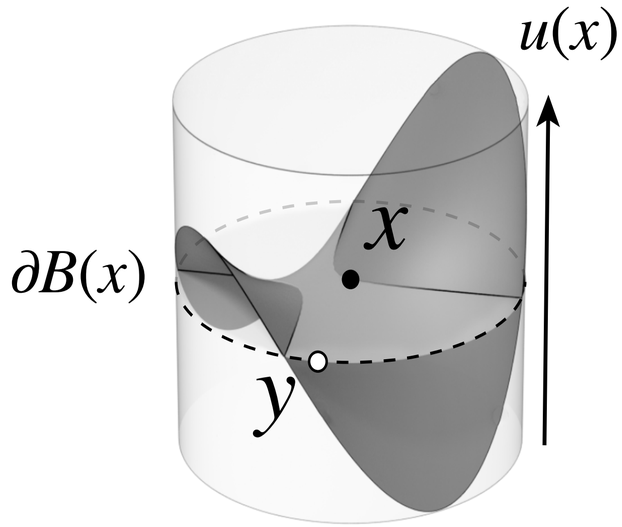
Q: Given this interpretation of the Laplacian, what can we say about the behavior of a ***harmonic function*** (i.e. a function u satisfying $\Delta u = 0$) ?

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

Not just an approximation, holds exactly 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?

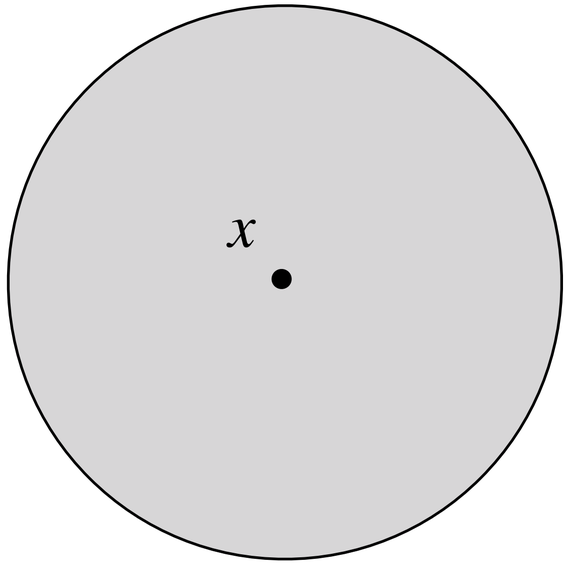
Monte Carlo estimation of the mean value integral

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

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



Monte Carlo estimation of the mean value integral

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

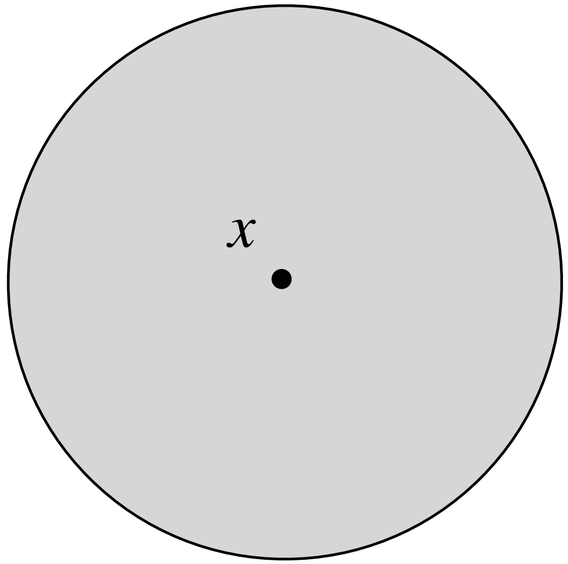
$$\begin{aligned} \Delta u &= 0 \quad \text{on } \Omega \\ u &= g \quad \text{on } \partial\Omega \end{aligned}$$

mean value property

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

mean value estimator

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



Monte Carlo estimation of the mean value integral

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

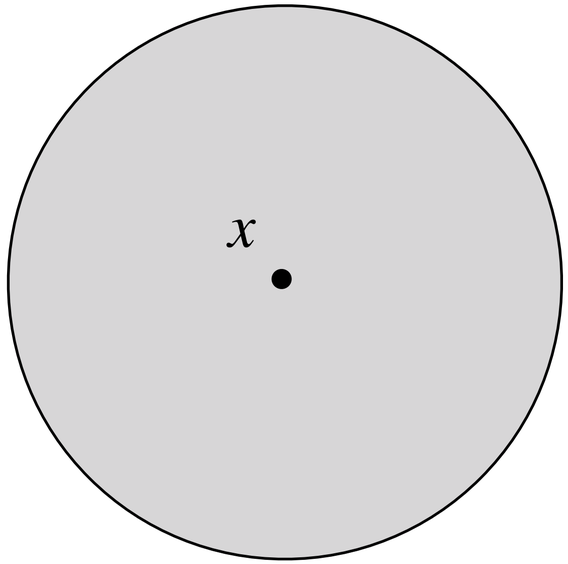
$$\begin{aligned} \Delta u &= 0 \quad \text{on } \Omega \\ u &= g \quad \text{on } \partial\Omega \end{aligned}$$

mean value property

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

mean value estimator

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



Monte Carlo estimation of the mean value integral

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

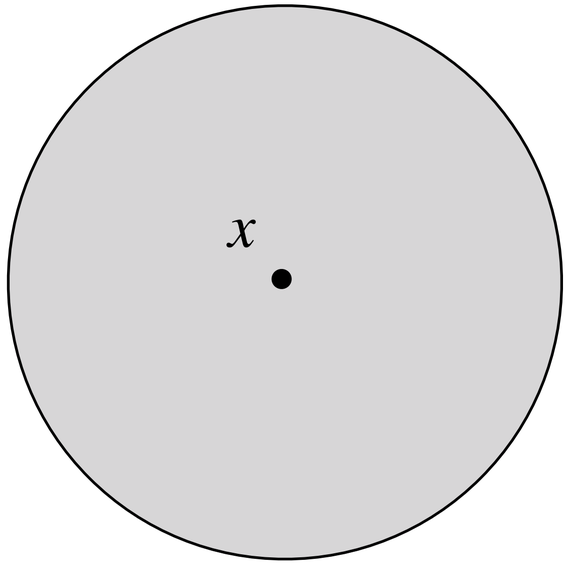
$$\begin{aligned} \Delta u &= 0 \quad \text{on } \Omega \\ u &= g \quad \text{on } \partial\Omega \end{aligned}$$

mean value property

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

mean value estimator

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



Monte Carlo estimation of the mean value integral

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

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

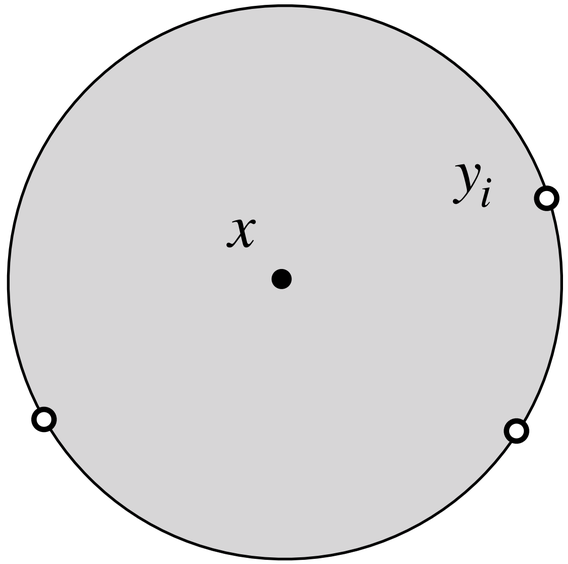
mean value property

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

mean value estimator

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

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

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

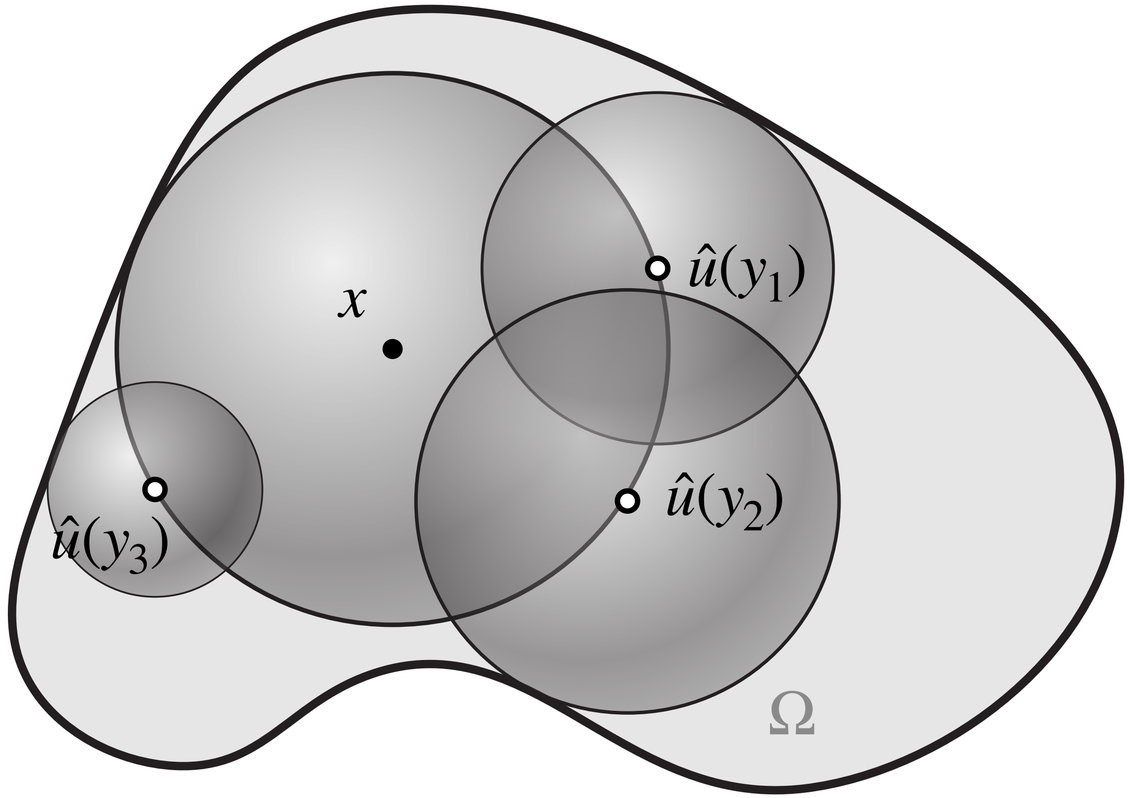
mean value property

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

mean value estimator

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

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



Branching Monte Carlo estimator

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

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

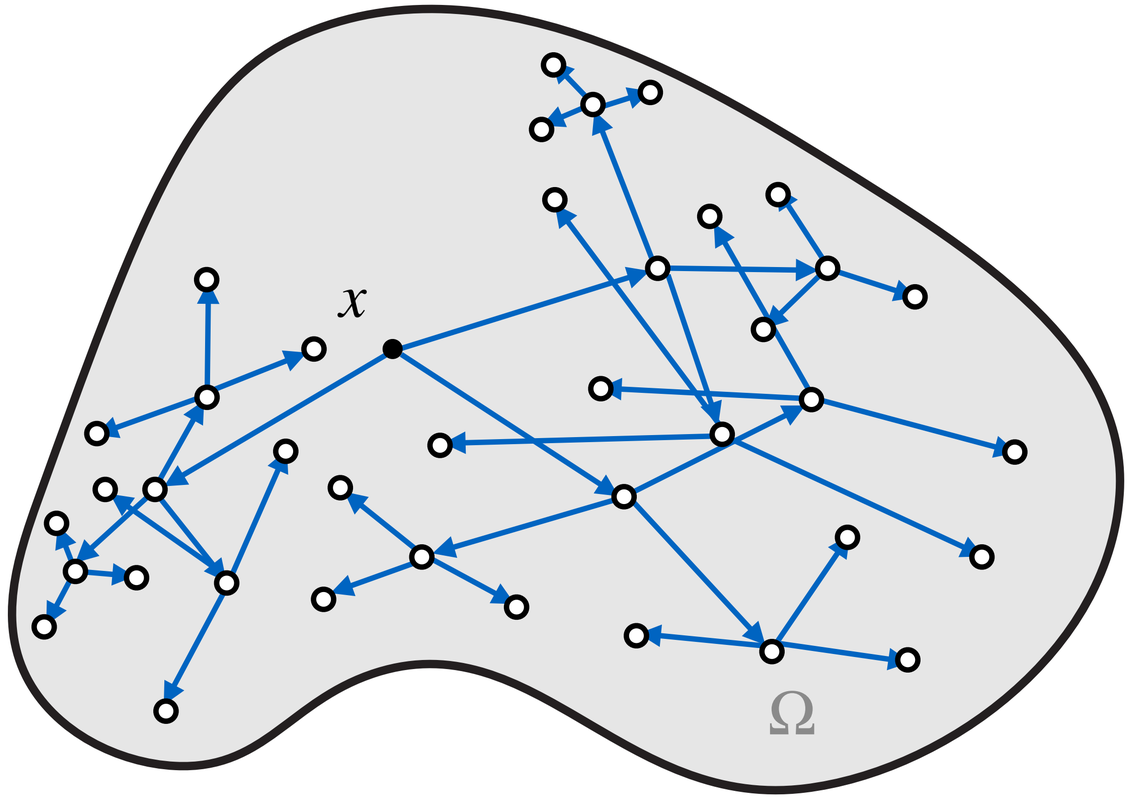
mean value property

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

mean value estimator

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

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



Branching Monte Carlo estimator

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

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

mean value property

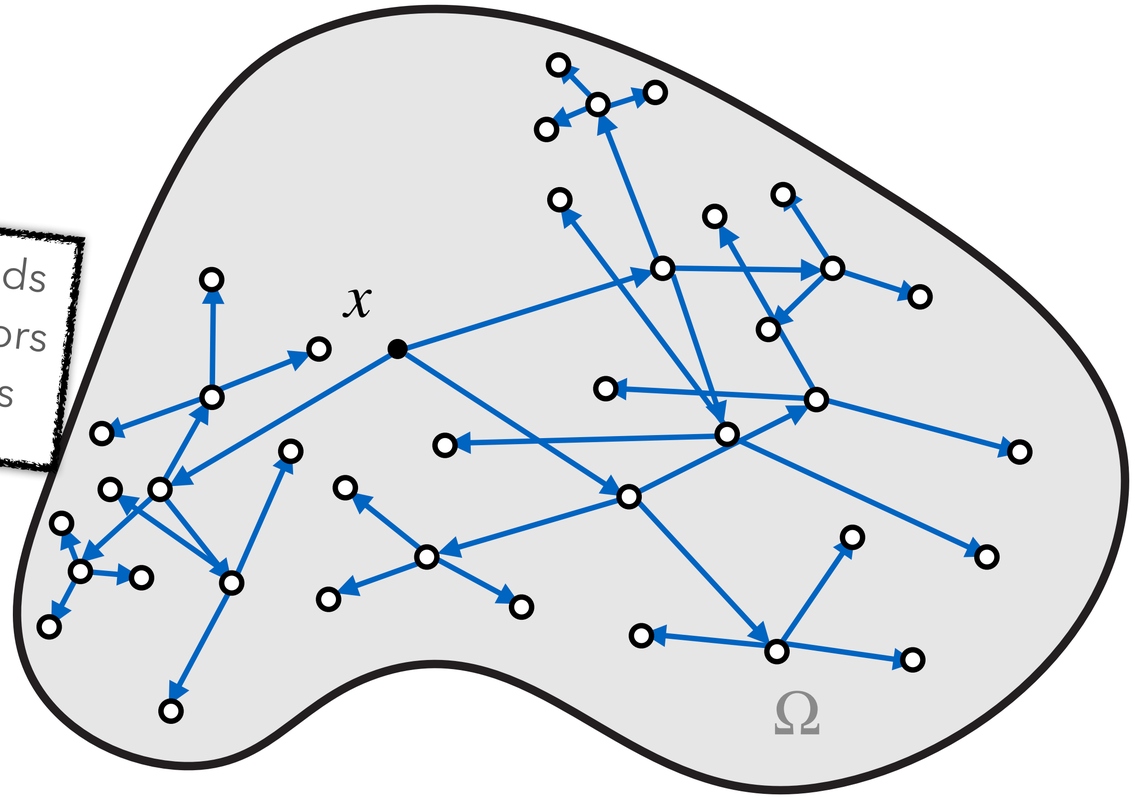
$$u(x) = \frac{1}{|\,\partial B(x)\,|} \int_{\partial B(x)} u(y)\; dy$$

mean value estimator

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

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

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



Walk on Spheres

To avoid branching, we only evaluate a single sample at each sphere

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

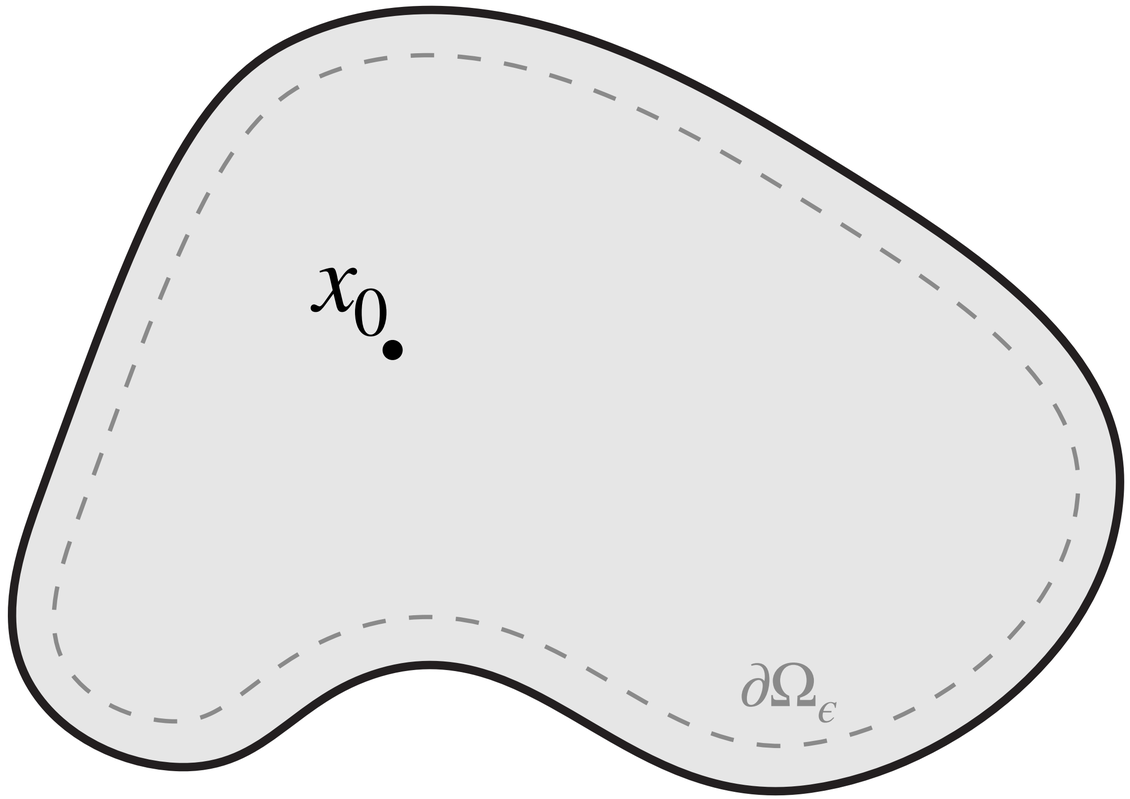
mean value property

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

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

To avoid branching, we only evaluate a single sample at each sphere

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

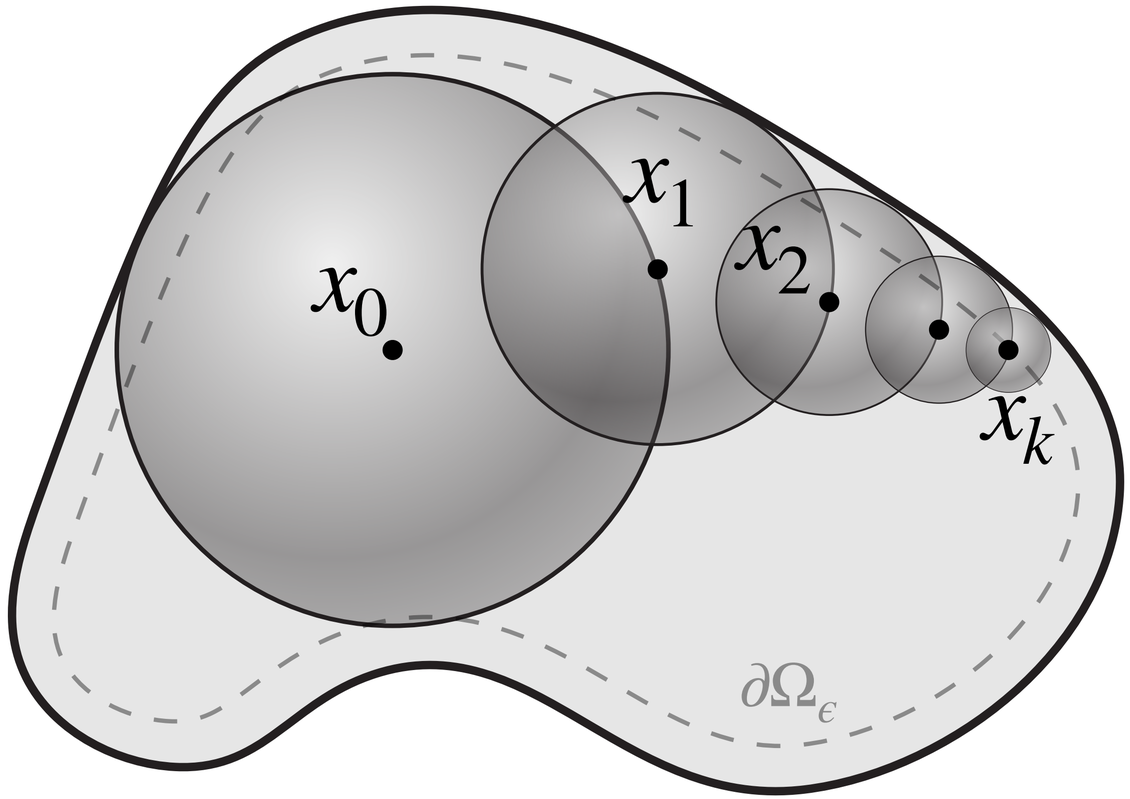
mean value property

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

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

To avoid branching, we only evaluate a single sample at each sphere

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

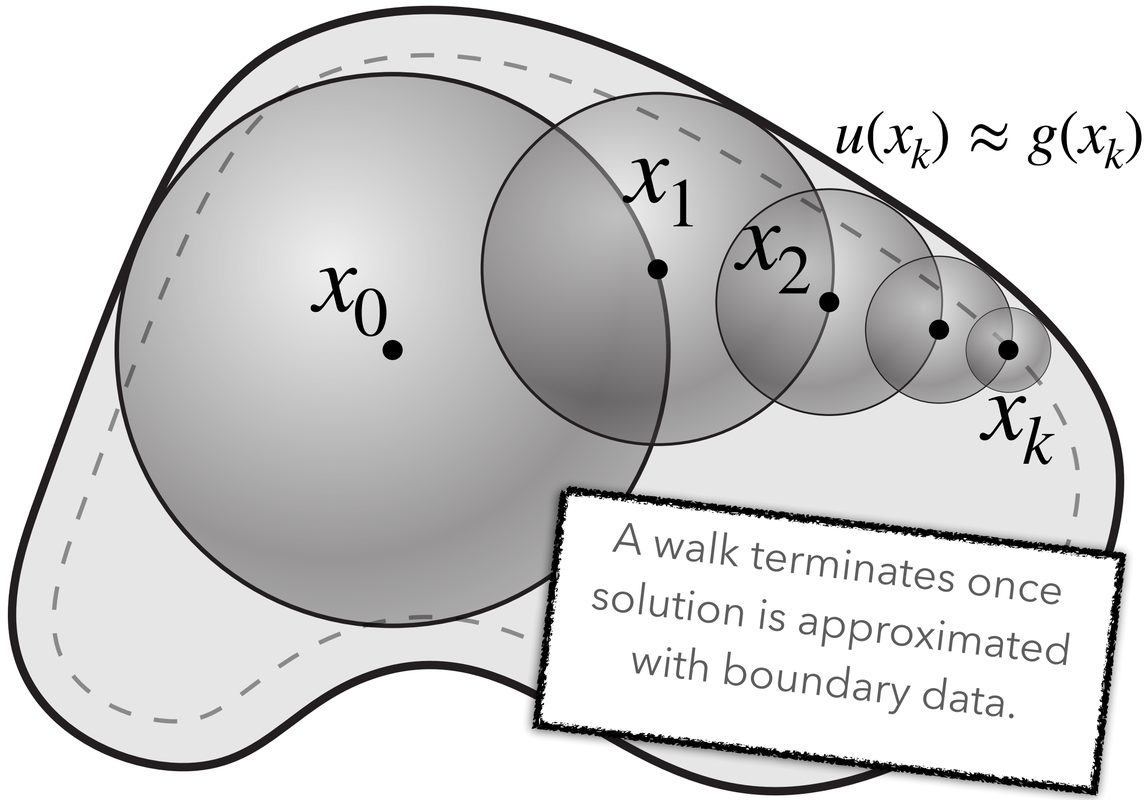
mean value property

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

walk on spheres estimator

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

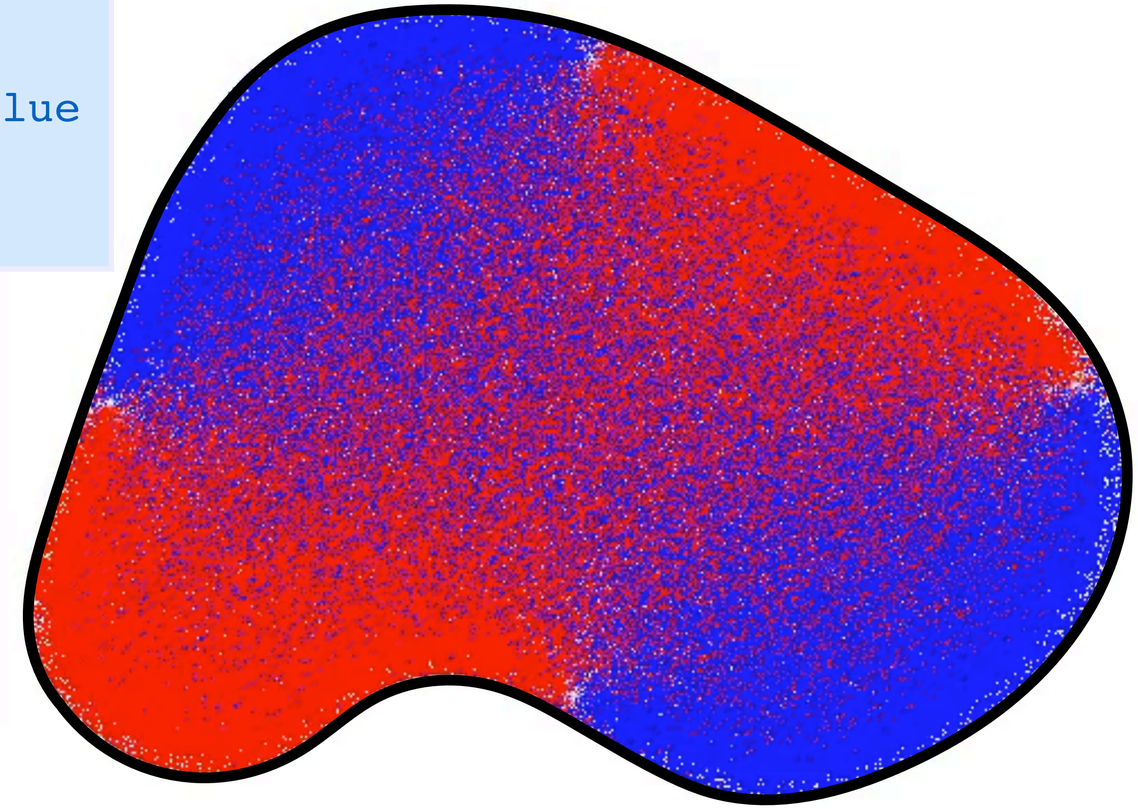
$$x_i \sim \mathcal{U}_{\partial B(x)}$$



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

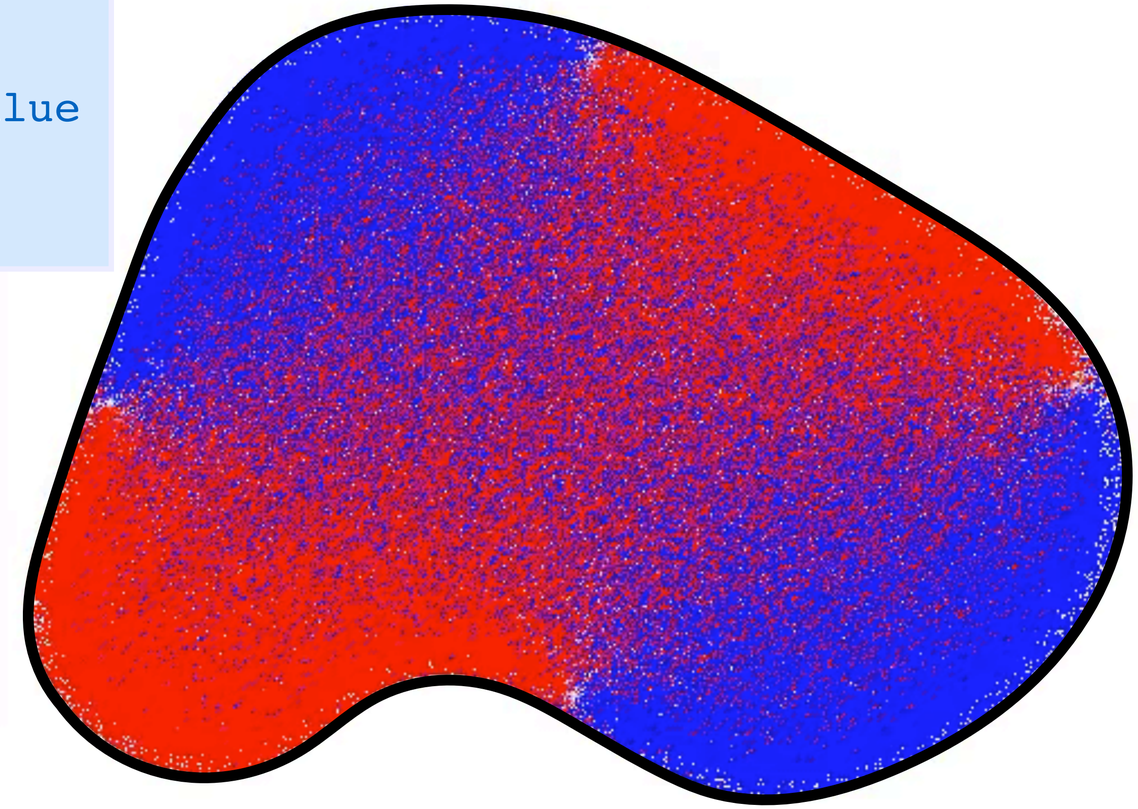
$$\hat{u}(x_i) = \begin{cases} \hat{u}(x_{i+1}) & \text{if } x_{i+1} \notin \partial\Omega_\epsilon \\ g(x_{i+1}) & \text{otherwise} \end{cases}$$

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

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
```

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



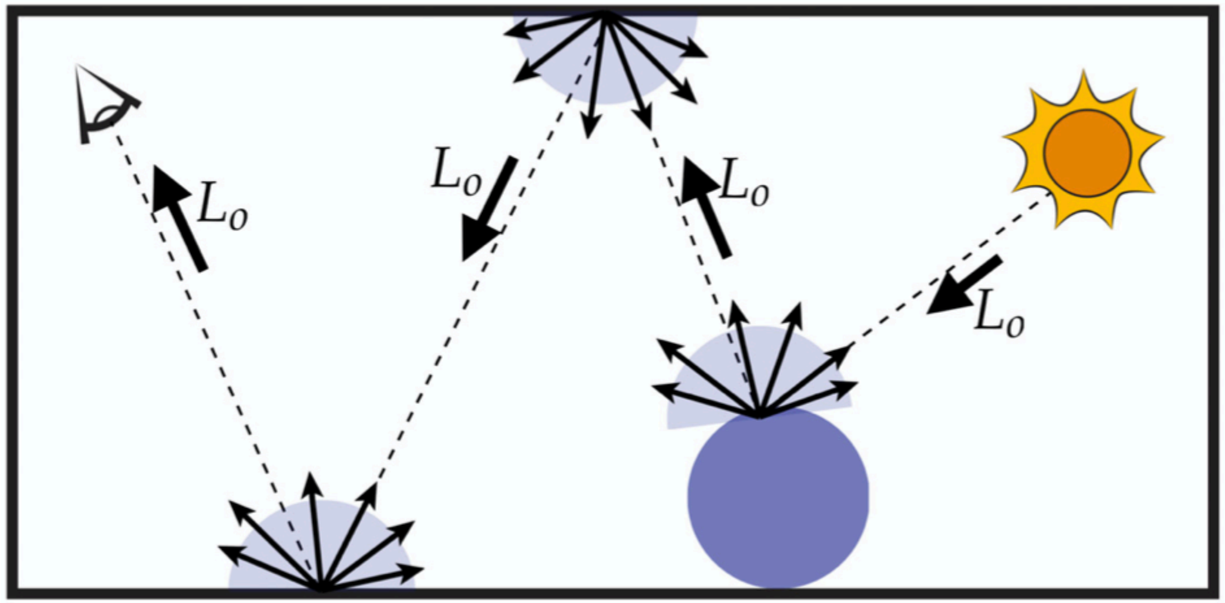
walk on spheres estimator

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

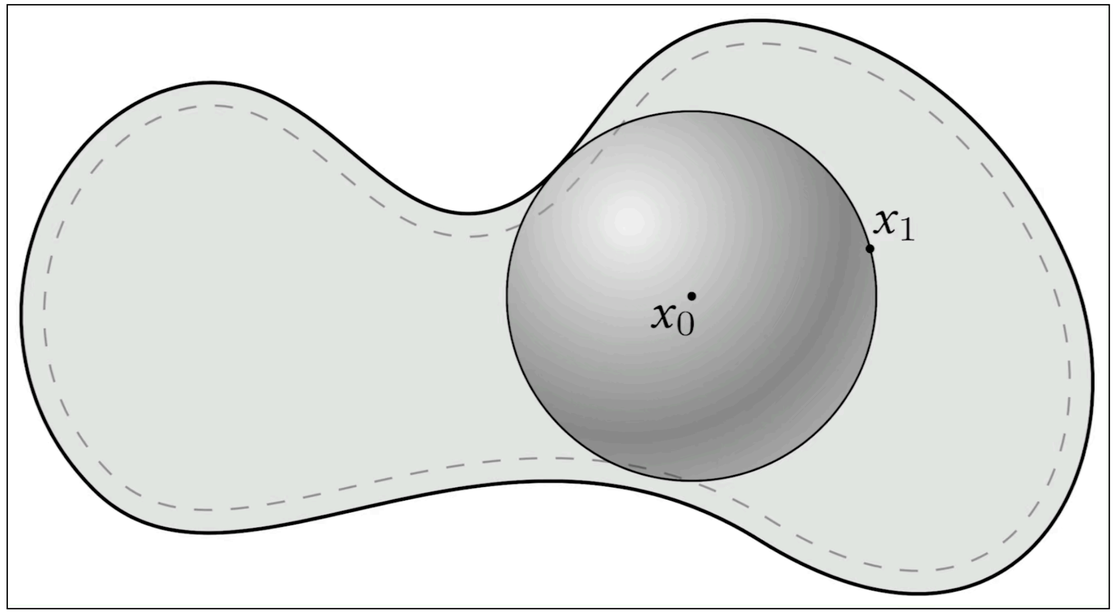
$$x_i \sim \mathcal{U}_{\partial B(x)}$$

Recursive estimators

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



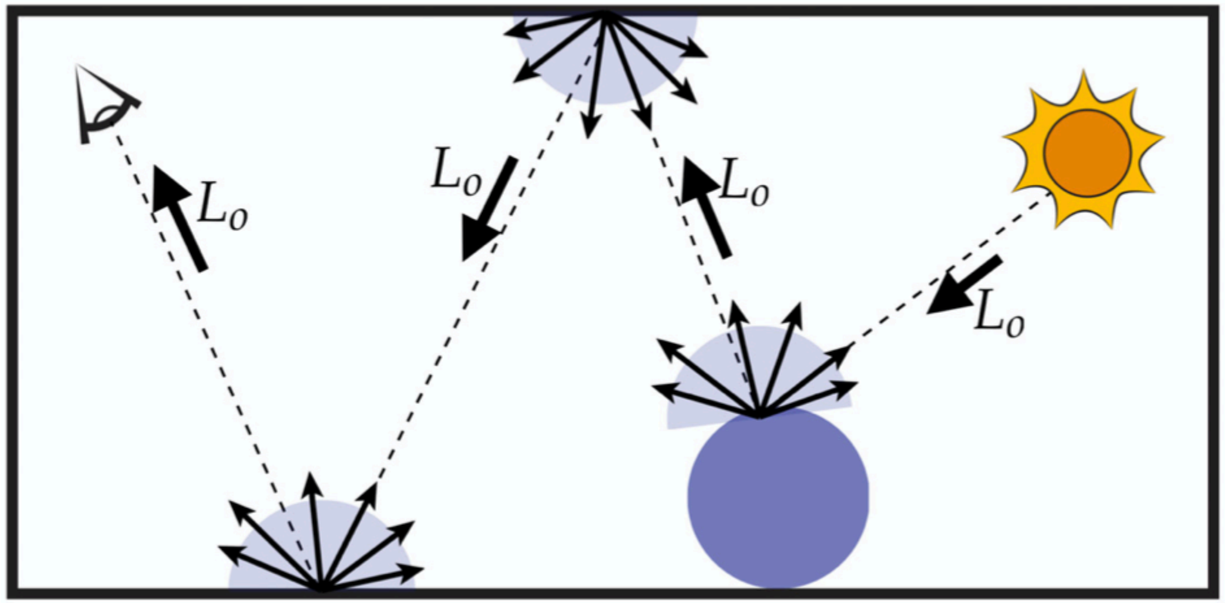
ray tracing



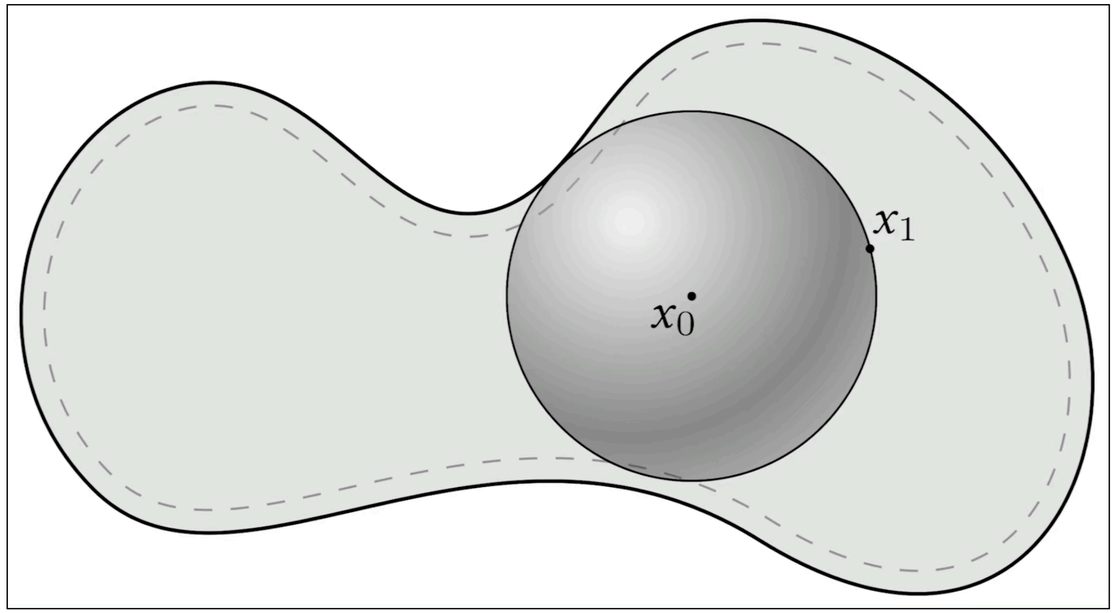
walk on spheres [Muller 1956]

Recursive estimators

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



ray tracing



walk on spheres [Muller 1956]

Warning about convergence

Warning about convergence

Earlier we showed that Monte Carlo estimators of integrals converge

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

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?

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

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 exists…

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

Convergence of WoS – integral viewpoint

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?

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

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

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 operator norm

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 operator norm

linear operator convergence

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

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 operator norm

linear operator convergence

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

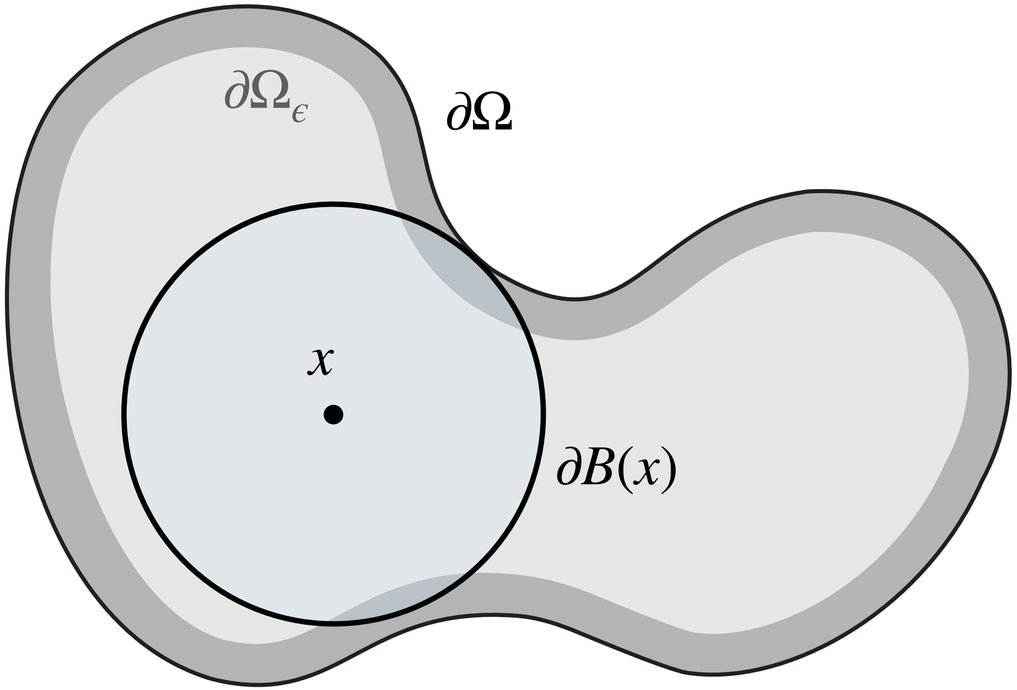
integral operator norm

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

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

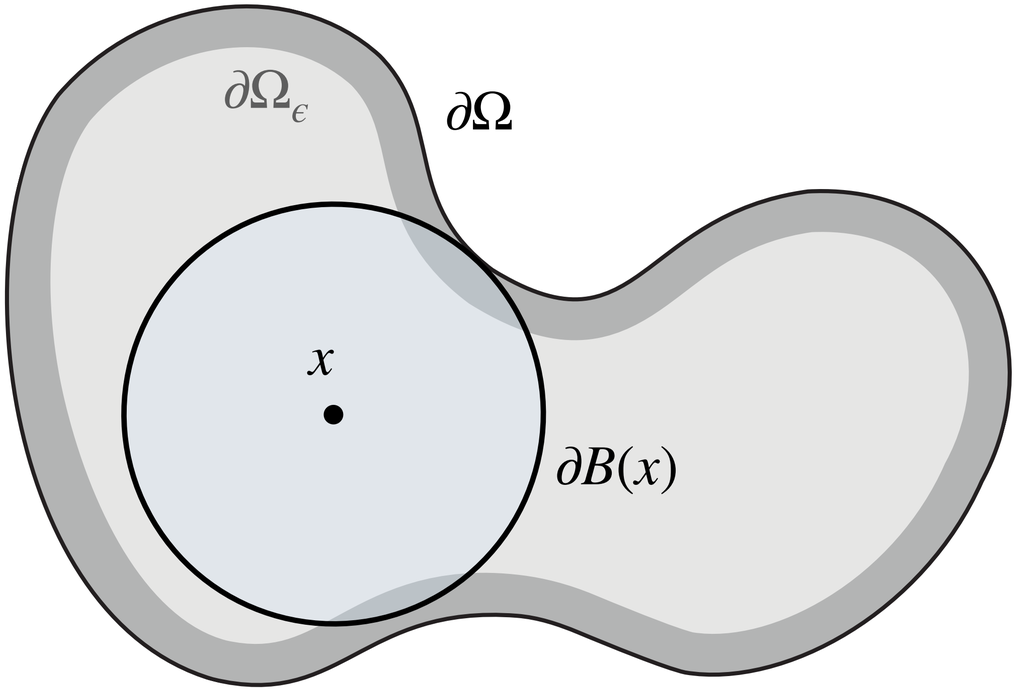
$$Lu(x) := \frac{1}{|\,\partial B(x)\,|}\int_{\partial B(x)} u(y)\ dy$$



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

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



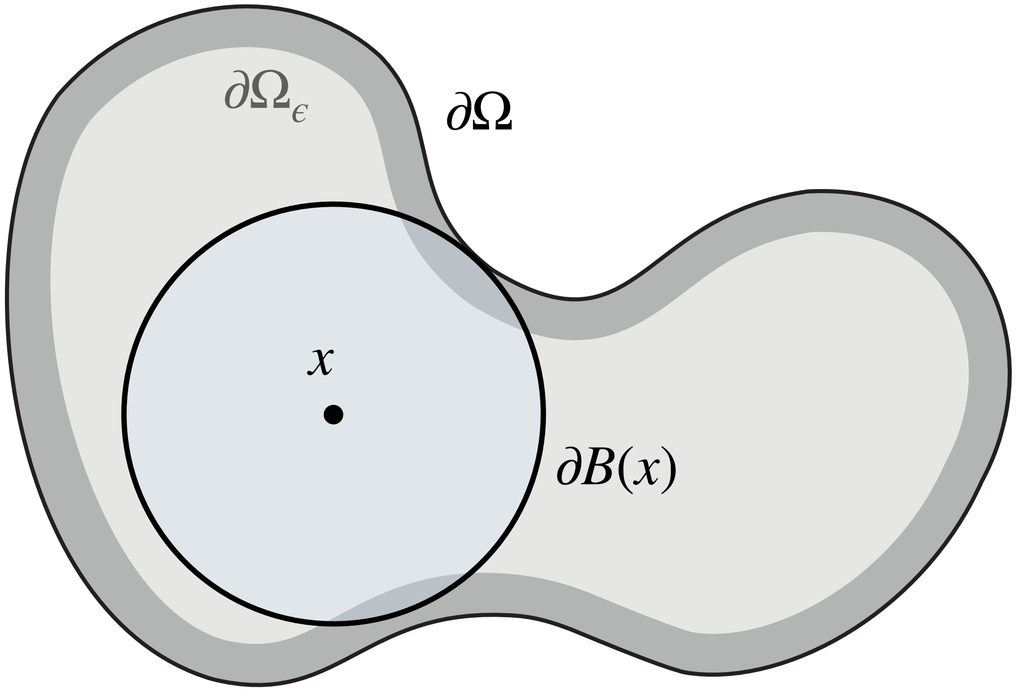
convergence criteria

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

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

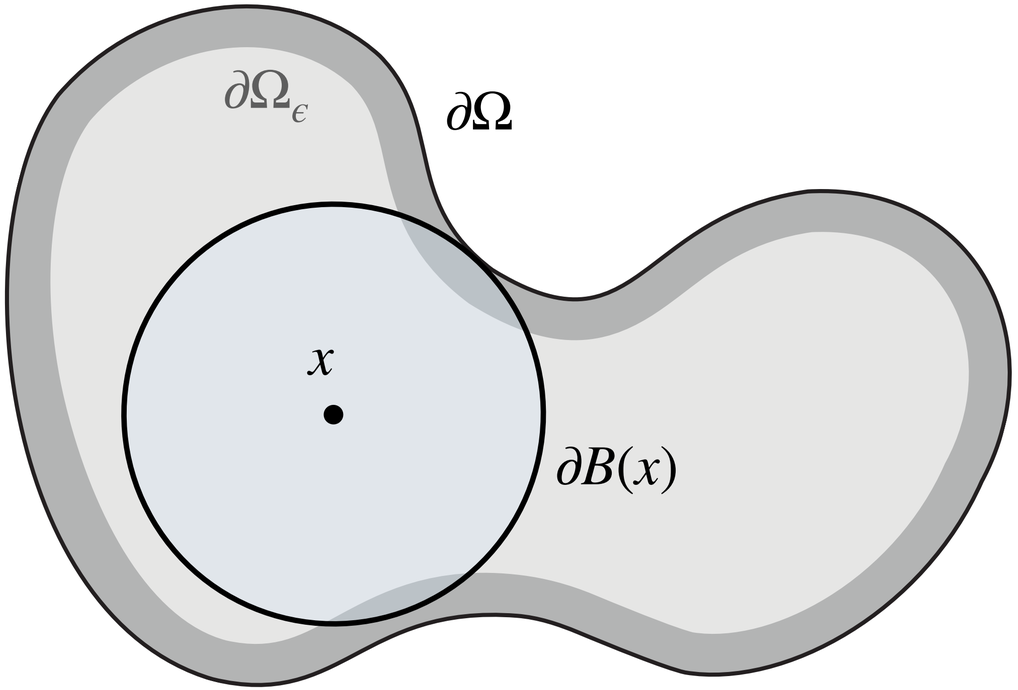
$$\|Lu(x)\|_{op} < \|u(x)\|_{op} \iff \|L\|_{\text{op}} < 1$$

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

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

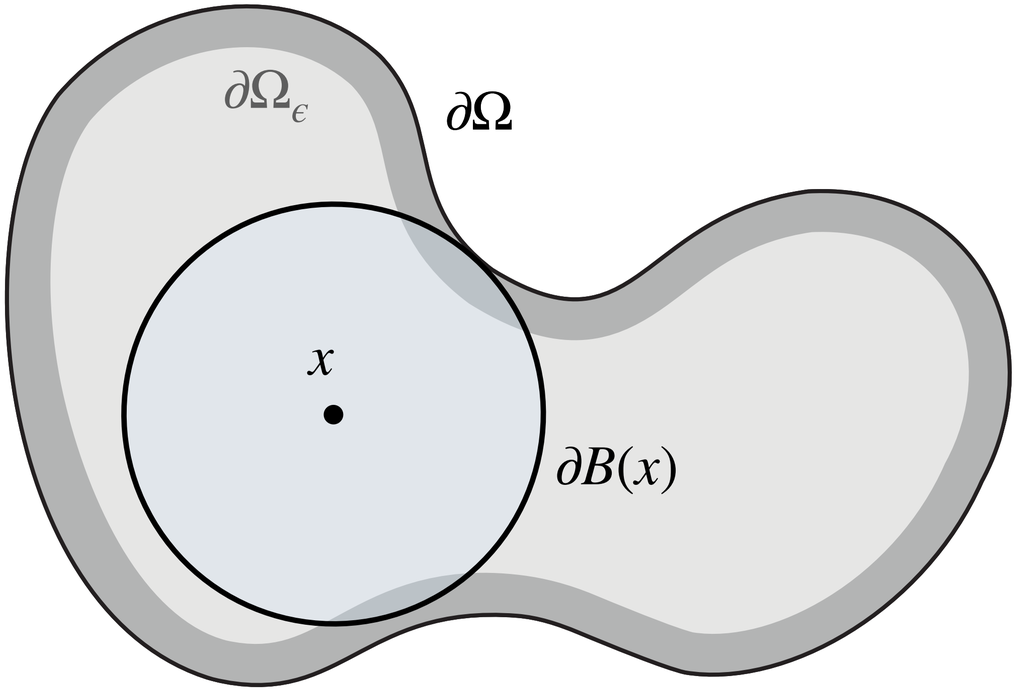
$$\|Lu(x)\|_{op} < \|u(x)\|_{op} \iff \|L\|_{\text{op}} < 1$$

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

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**

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



convergence criteria

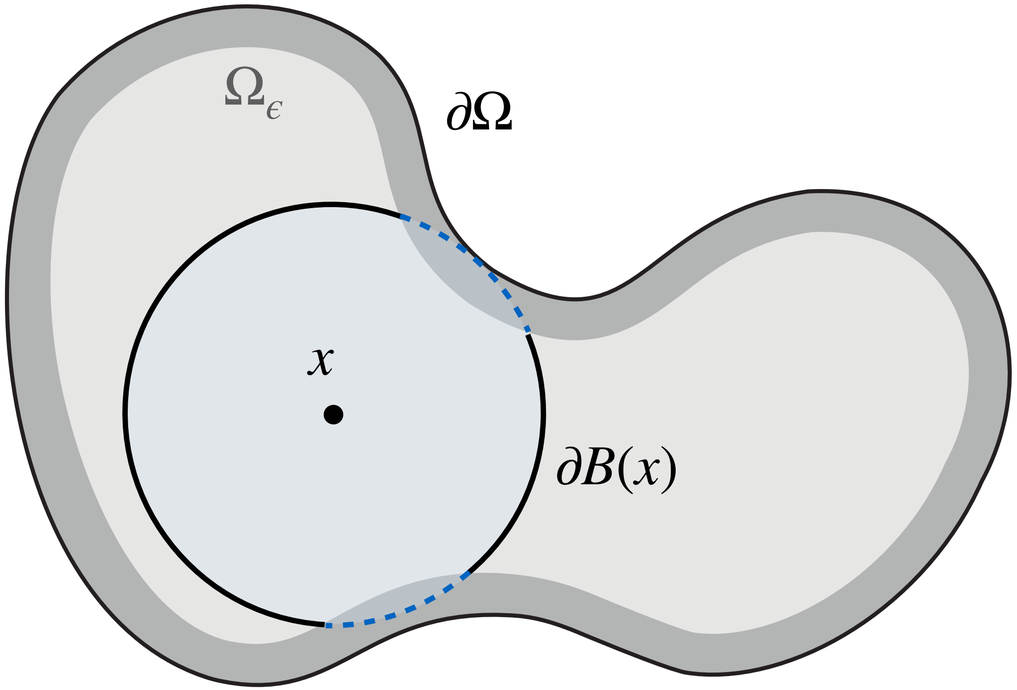
$$\|Lu(x)\|_{op} < \|u(x)\|_{op} \iff \|L\|_{\text{op}} < 1$$

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

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**

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



convergence criteria

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

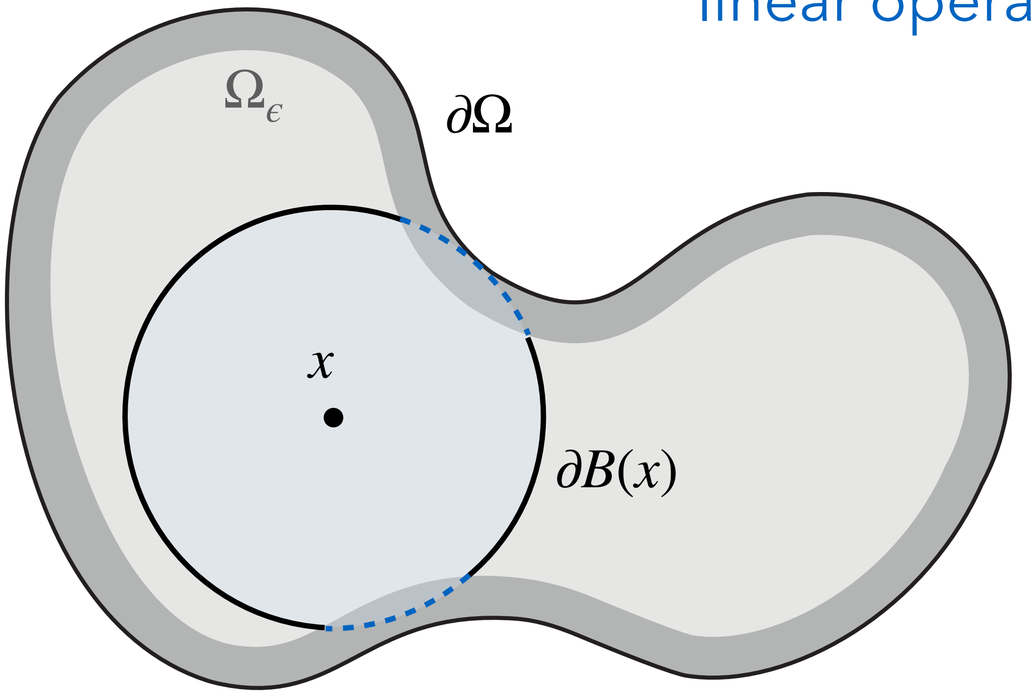
$$\|L\|_{\text{op}} = \frac{1}{|\,\partial B(x)\,|}\int_{\partial B(x)} \mathrm{d}x = 1$$

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

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

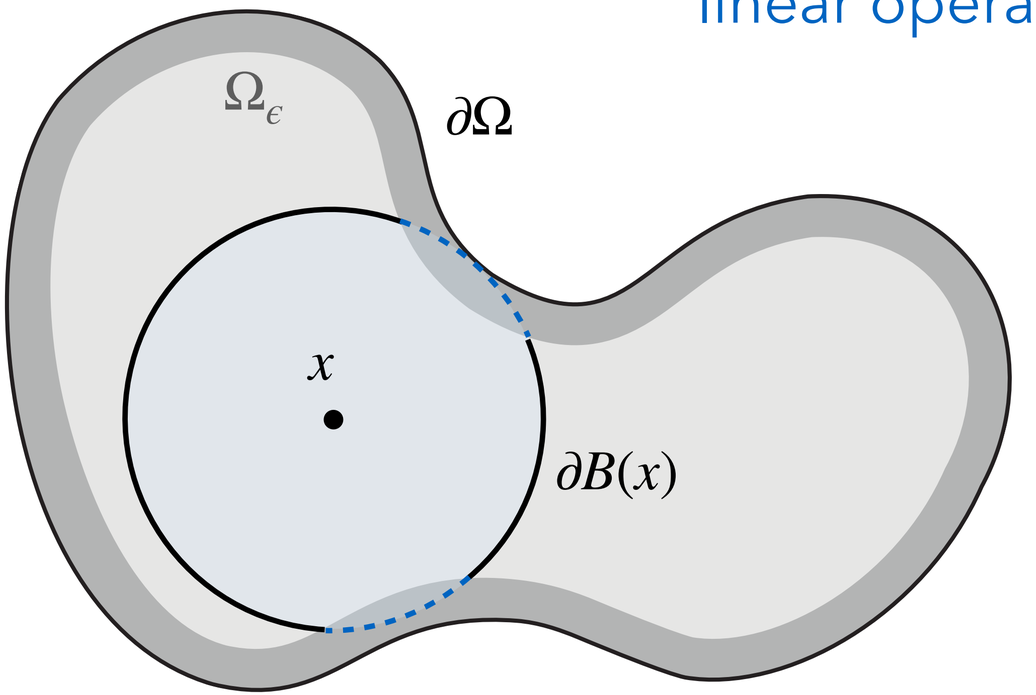
$$\|L\|_{\text{op}} = \frac{1}{|\,\partial B(x)\,|}\int_{\partial B(x)} \mathrm{d}x = 1$$

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

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

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

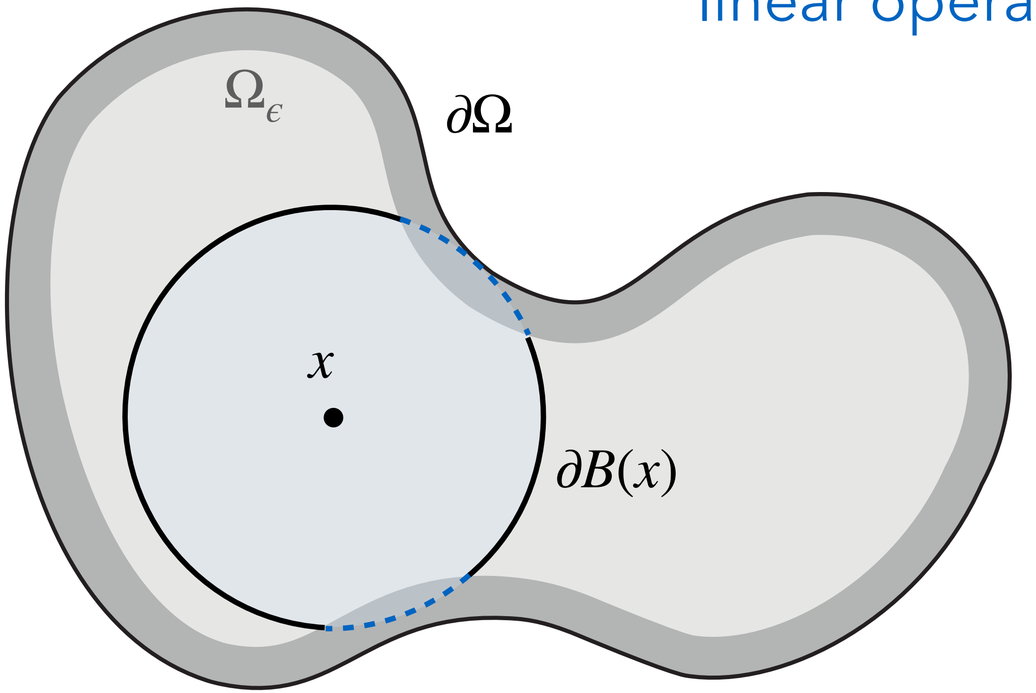
$$\|L_\epsilon\|_{\text{op}} = \frac{1}{|\,\partial B(x)\,|}\int_{\partial B(x)\setminus\Omega_\epsilon} \mathrm{d}x < 1$$

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

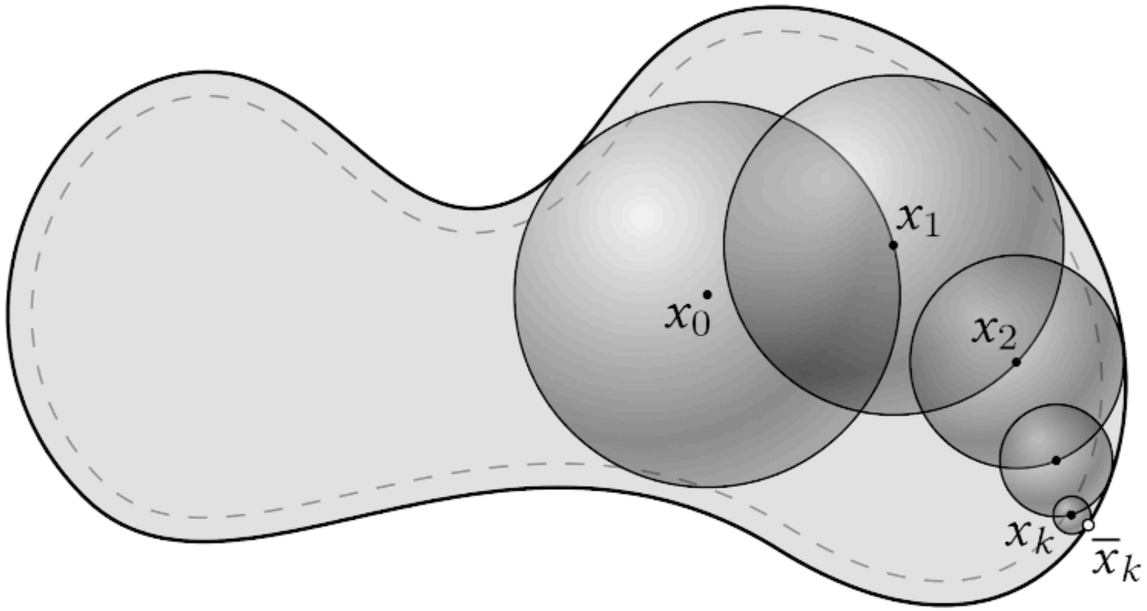
$$\|Lu(x)\|_{op} < \|u(x)\|_{op} \iff \|L\|_{\text{op}} < 1$$

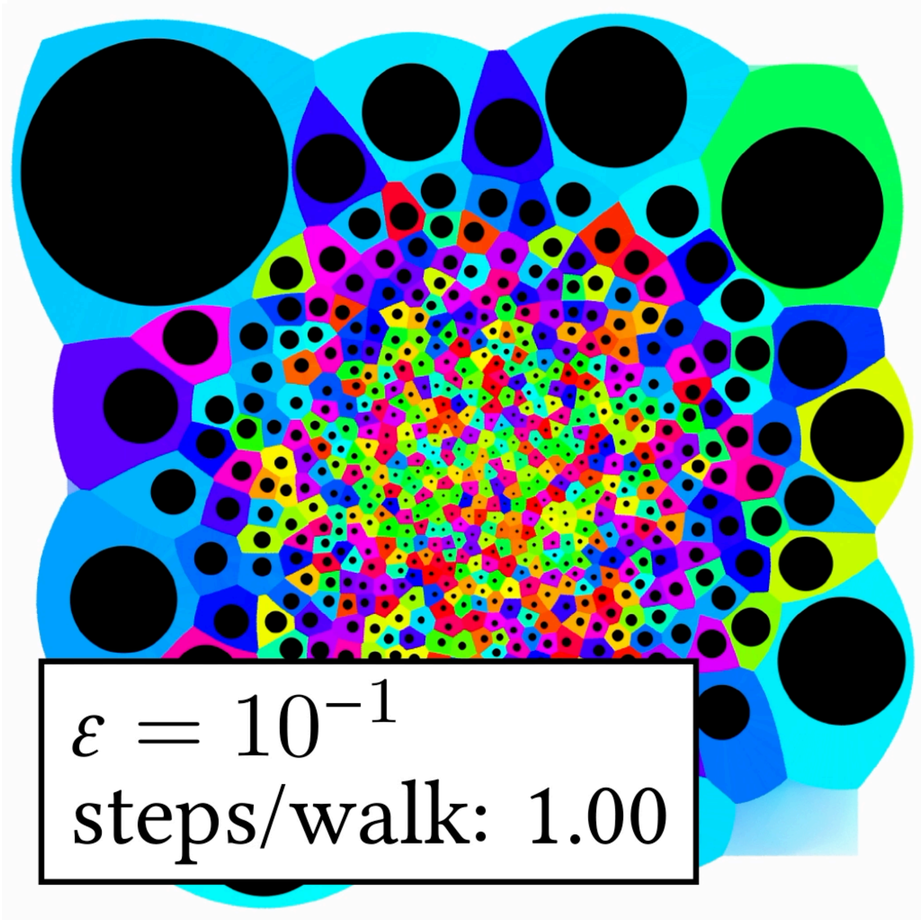
$$\|L\|_{\text{op}} = \frac{1}{|\,\partial B(x)\,|}\int_{\partial B(x)} \mathrm{d}x = 1$$

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

Stopping tolerance ε for Dirichlet boundary

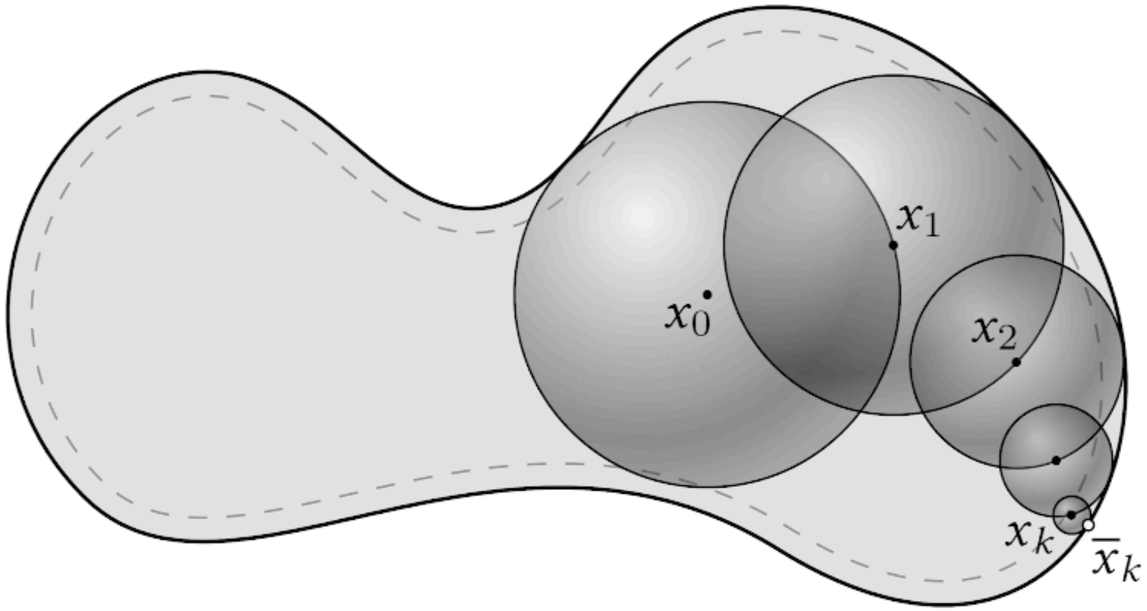
Introduces minimal bias and has little impact on performance

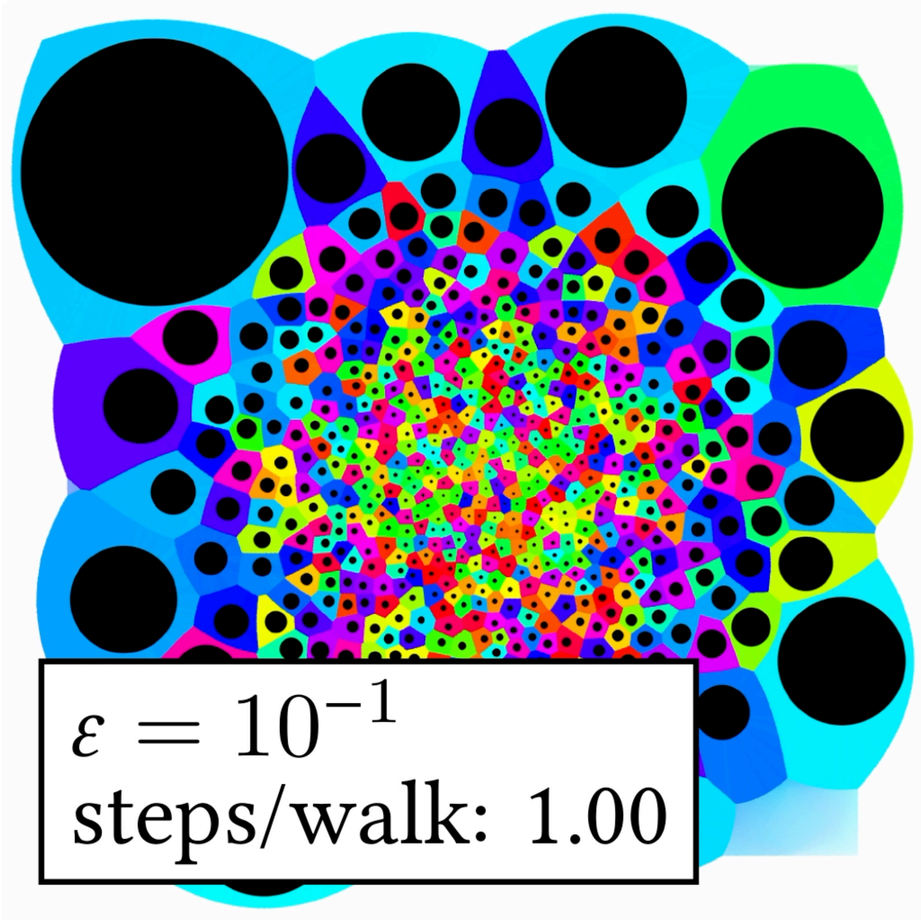




Stopping tolerance ε for Dirichlet boundary

Introduces minimal bias and has little impact on performance





Efficiency of Walk on Spheres

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:*

	Rate of convergence
$\alpha < 2$	$O\big(\log 1/\varepsilon\big)$
$\alpha = 2$	$O\big(\log^2 1/\varepsilon\big)$
$\alpha > 2$	$O\big((1/\varepsilon)^{2-4/\alpha}\big)$

(3)

The $O(\cdot)$ in the expressions above depends on the dimension d, on α, on the thickness constant C from Definition 1 and on $\beta > 0$ from the definition of the WoS. It does not depend directly on Ω or on the choice of initial point X_0.

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)

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.

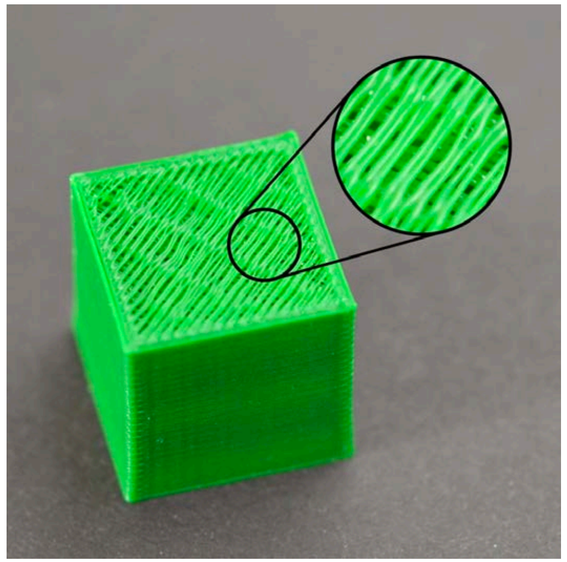
does not depend directly on [illegible]

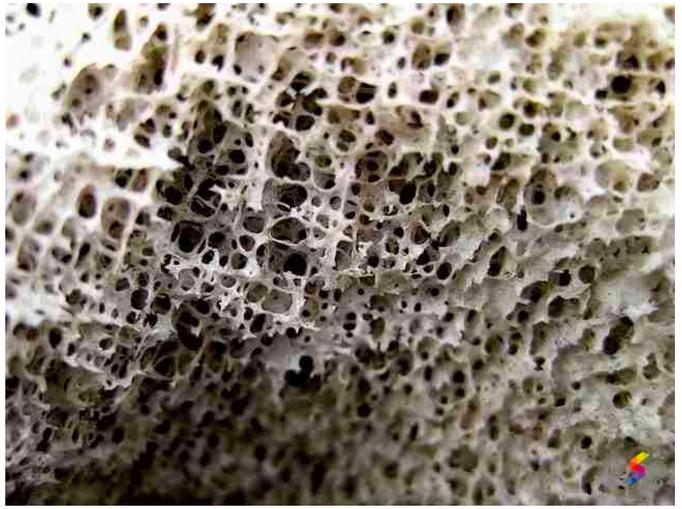
Moreover, the rates of convergence above are tight. Formally, let $f_\alpha(1/\varepsilon)$ be the growth rate asymptotically given by the formulas in (3). For any α and for any $g(1/\varepsilon)$ such that $g(1/\varepsilon) = o(f_\alpha(1/\varepsilon))$, there is an α-thick domain Ω_g^α with some thickness 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.

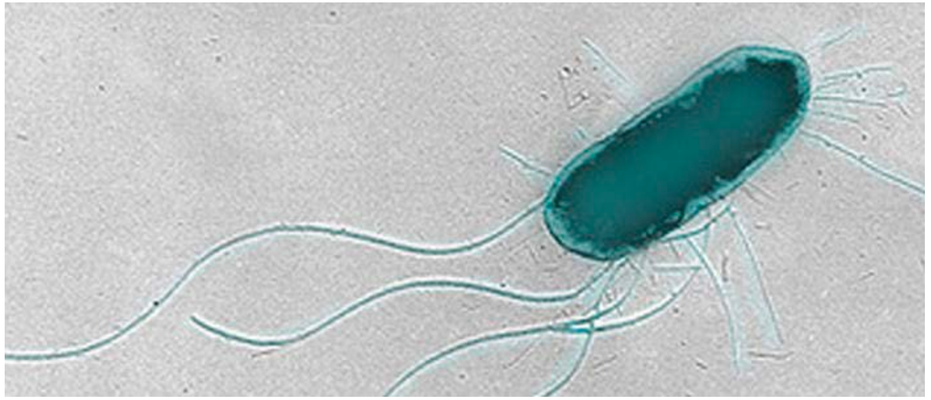
Binder & Braverman, "The Rate of Convergence of the Walk on Spheres Algorithm" (2012)

Stochastic vs. Deterministic Methods

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

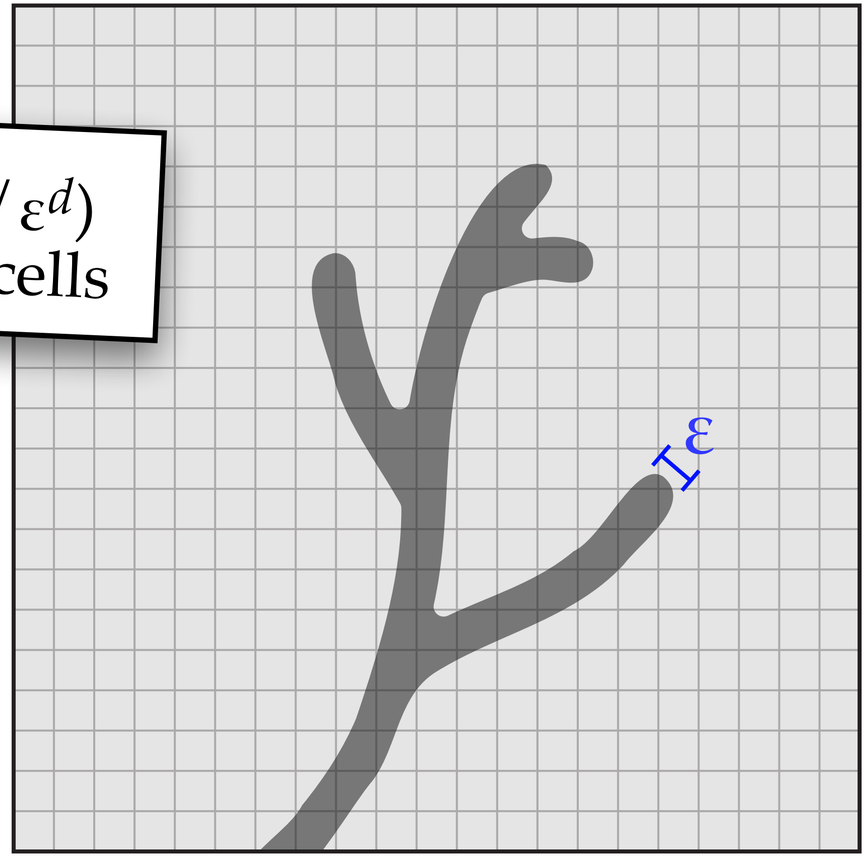




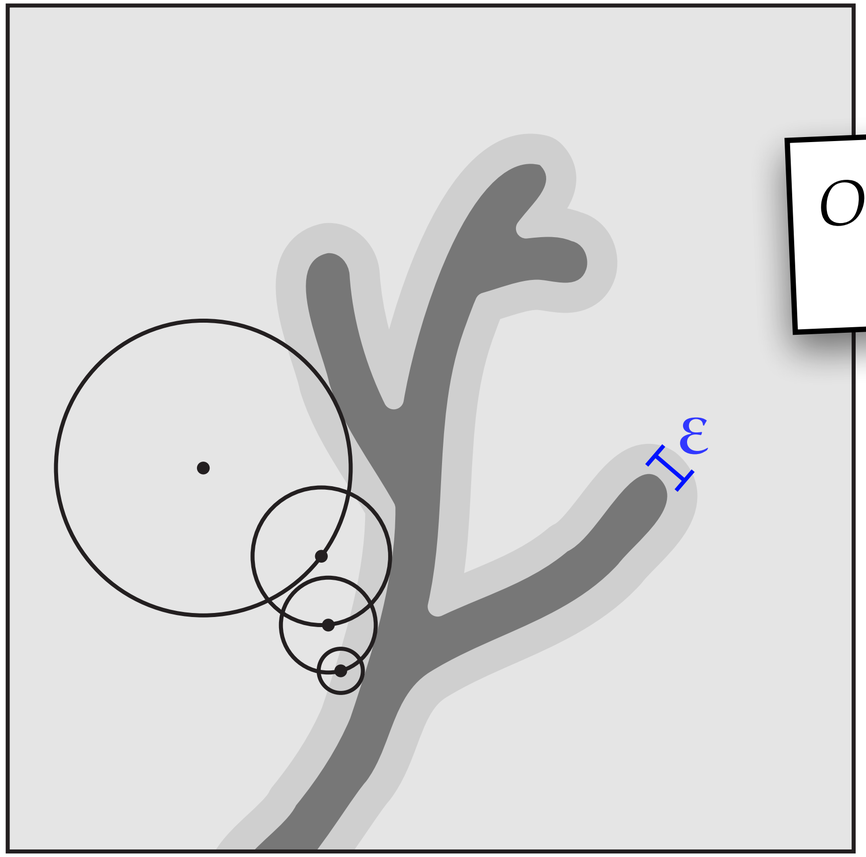


Stochastic vs. Deterministic Methods

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



finite differences

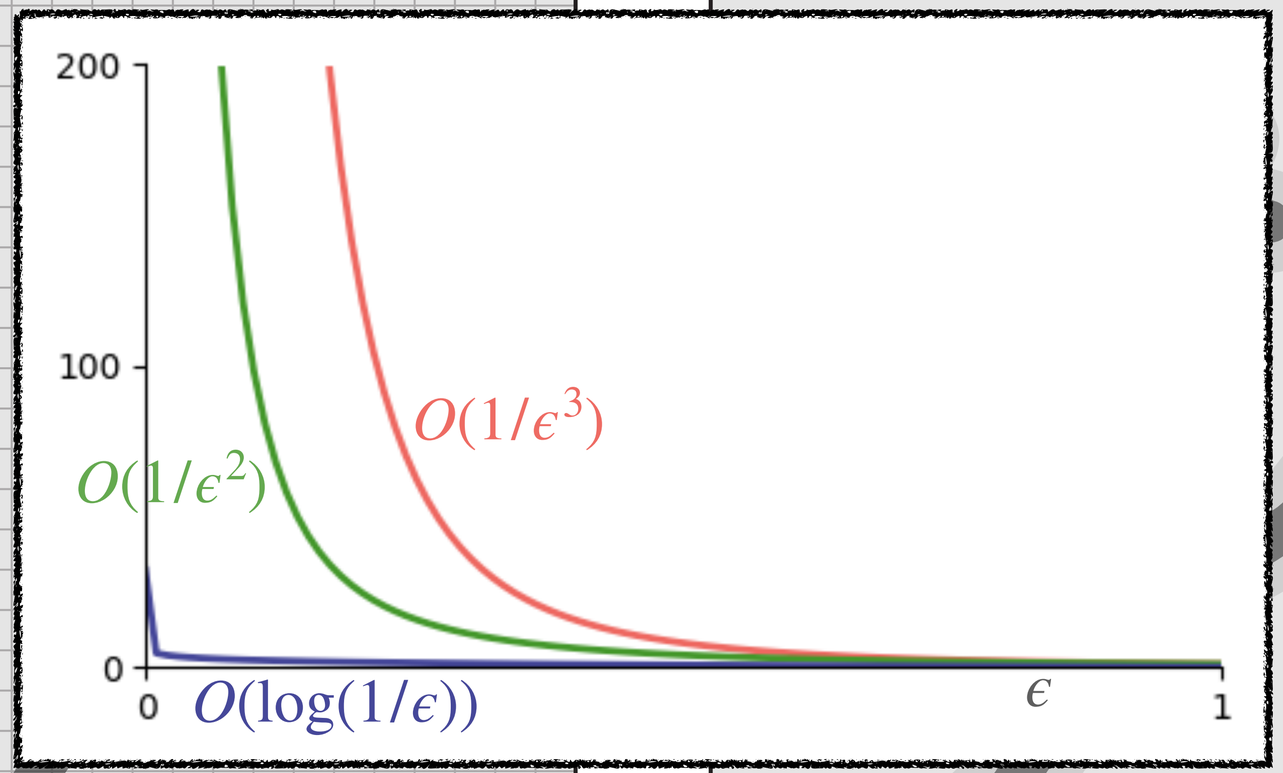


$O(\log 1/\varepsilon)$ steps

walk on spheres

Stochastic vs. Deterministic Methods

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



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

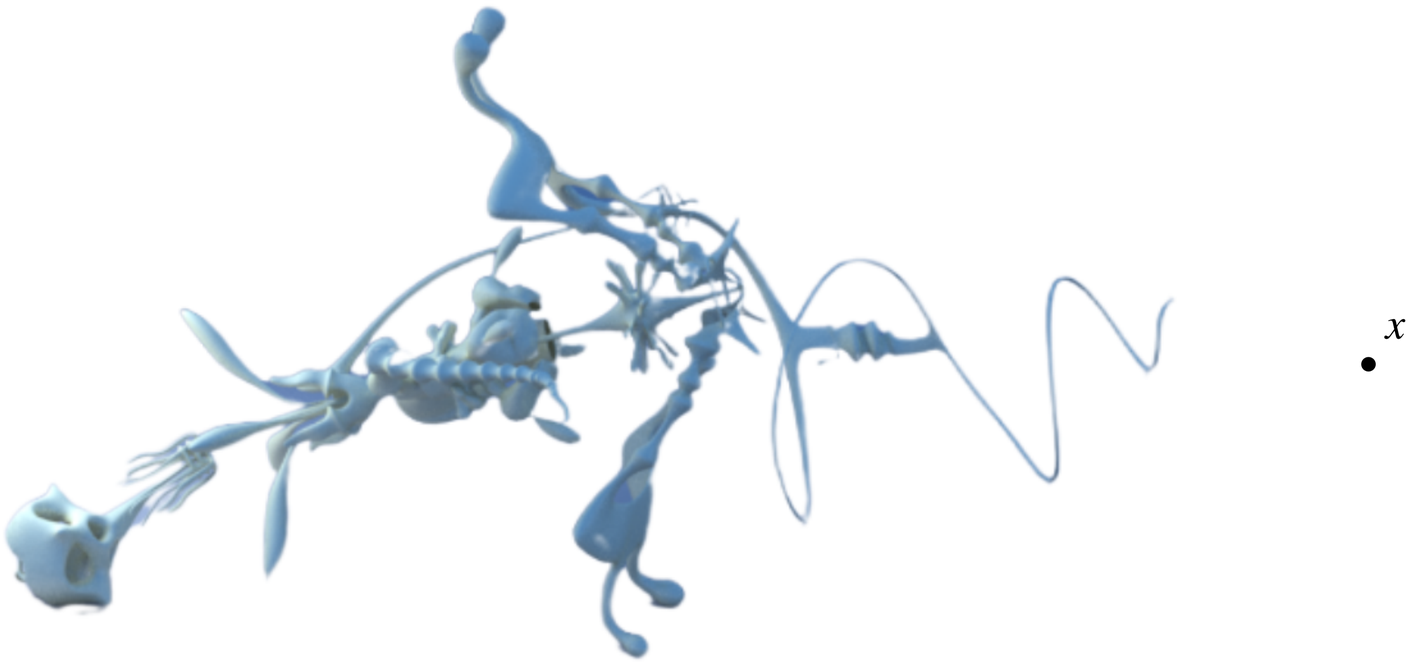
ε

finite differences

walk on spheres

Closest Point Queries

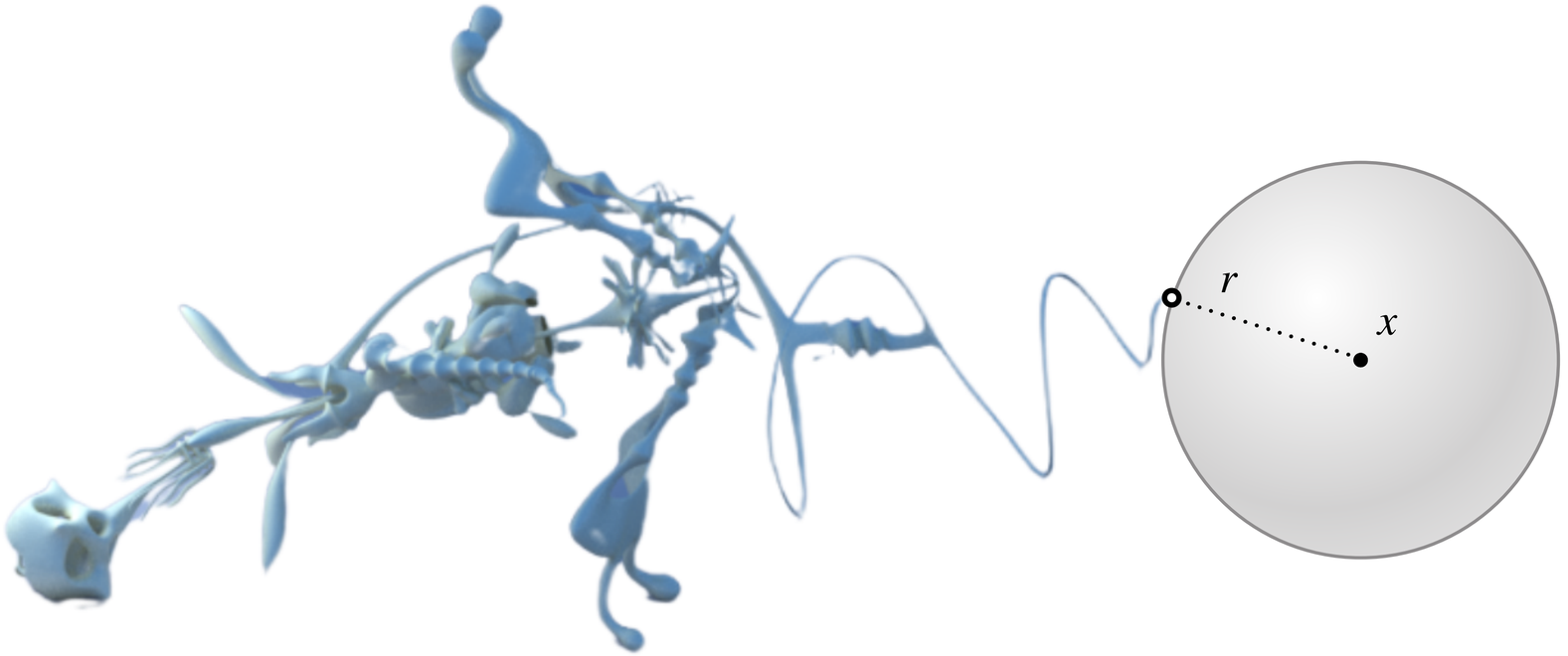
Q: How expensive is a single step of walk on spheres?



Closest Point Queries

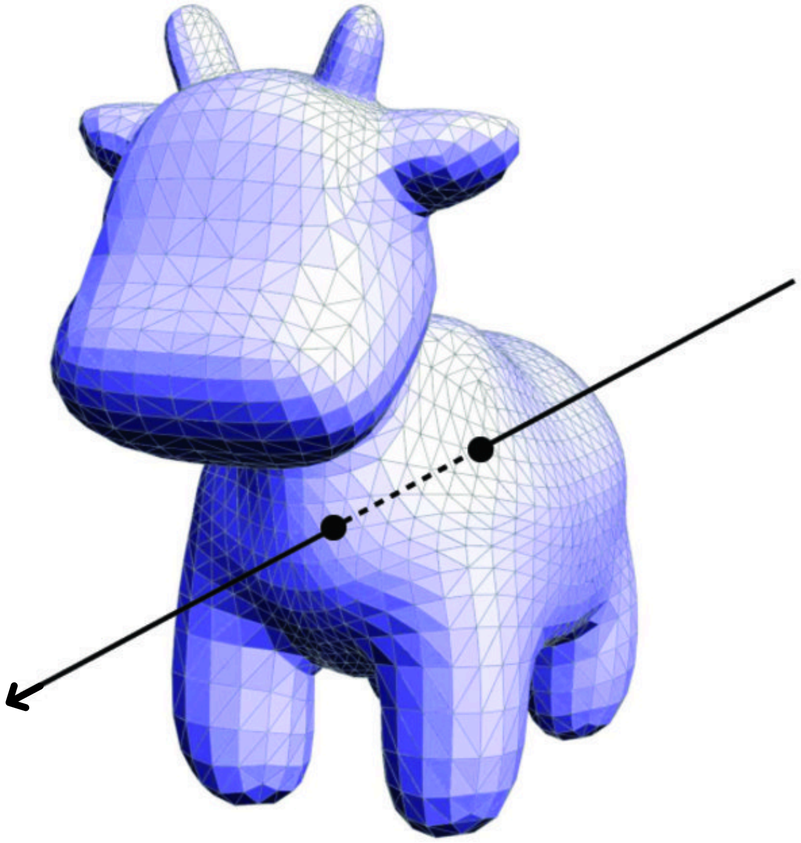
Q: How expensive is a single step of walk on spheres?

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



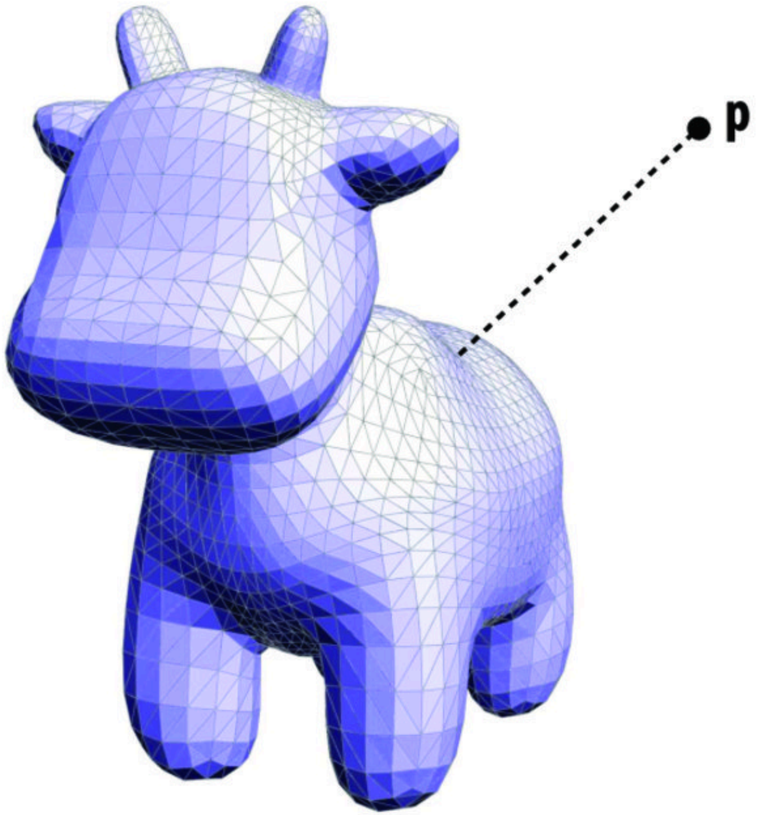
Geometric queries in Monte Carlo methods

rendering



Ray Intersection Query

walk on spheres

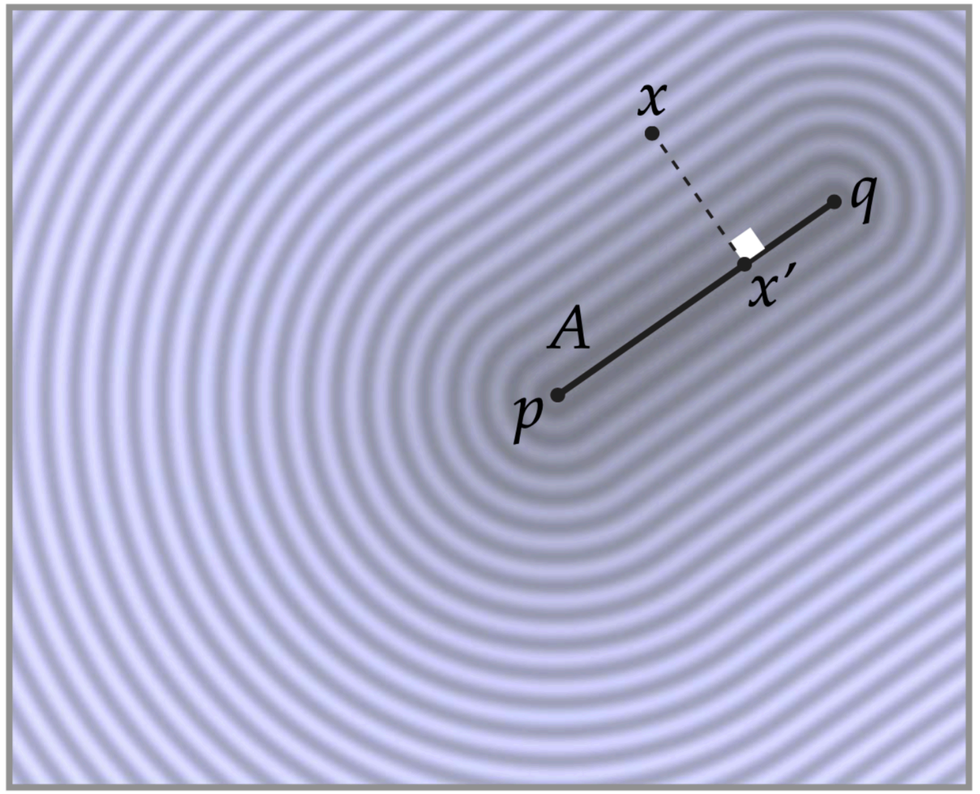


Distance Query

Closest point queries

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

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



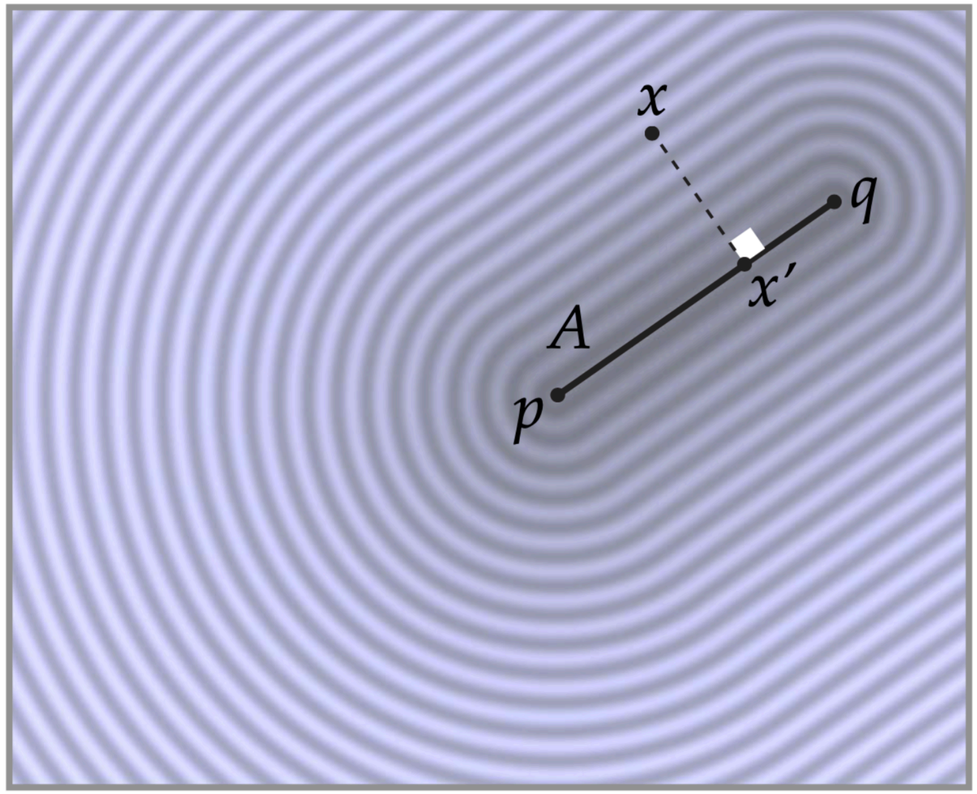
Closest point queries

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

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

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

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



Closest point queries

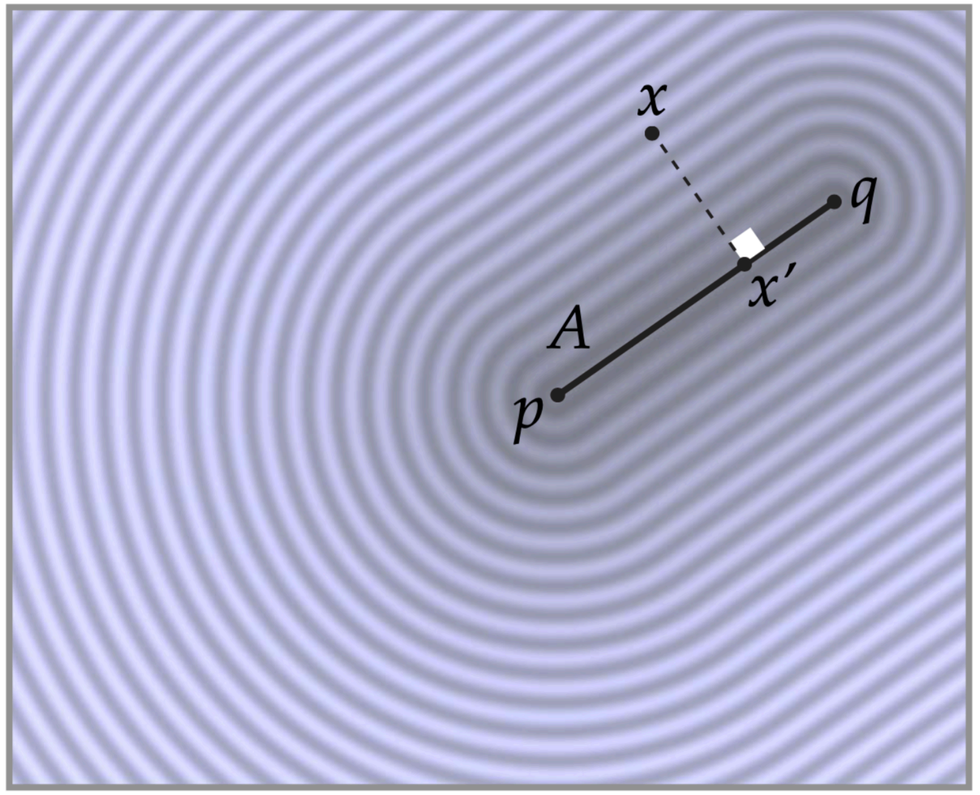
For query point x, find closest point x' on domain boundary $\partial\Omega$

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

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

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

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



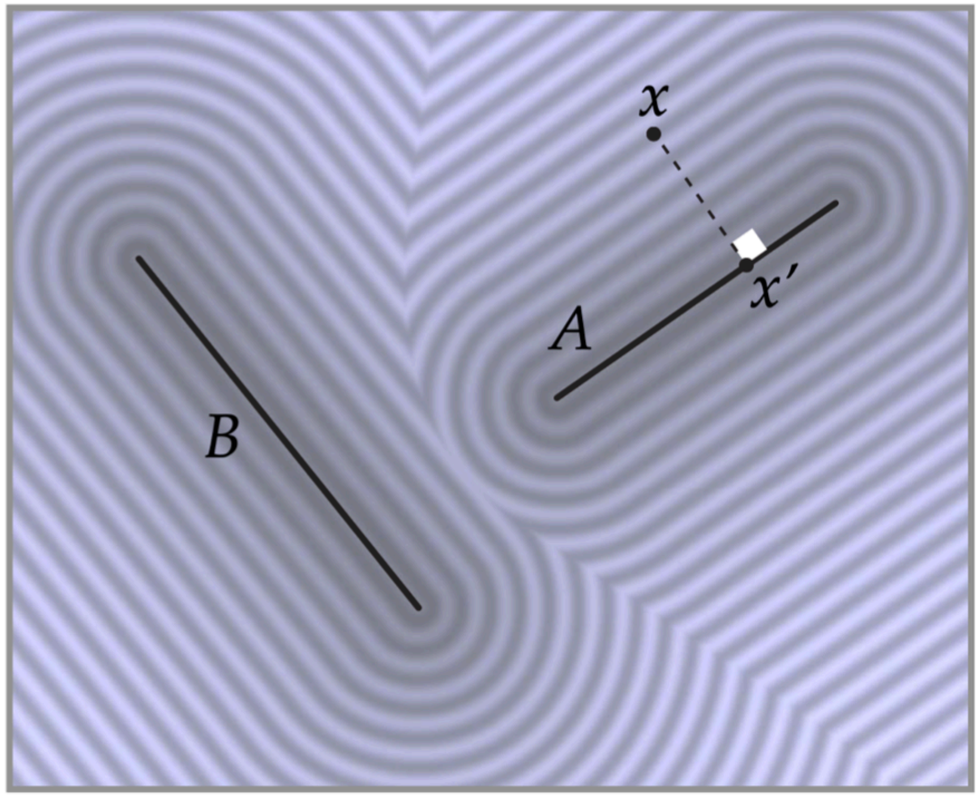
Closest point queries

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 ($\partial\Omega = A \cup B$)



Closest point queries

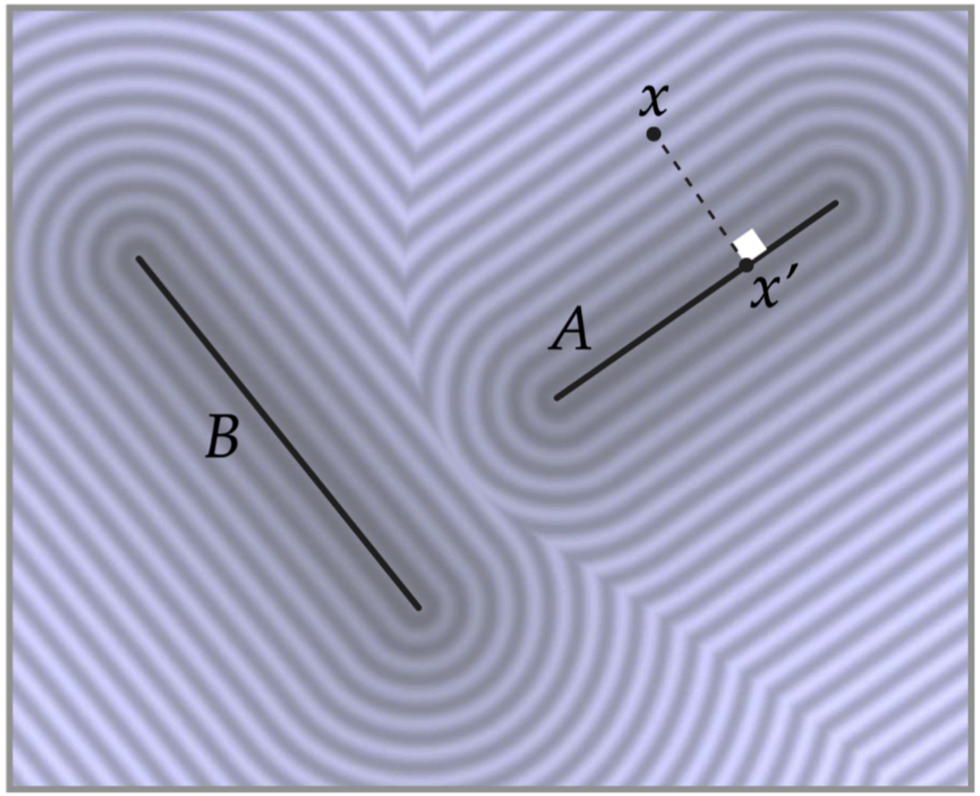
For query point x, find closest point x' on domain boundary $\partial\Omega$

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 $\partial\Omega$

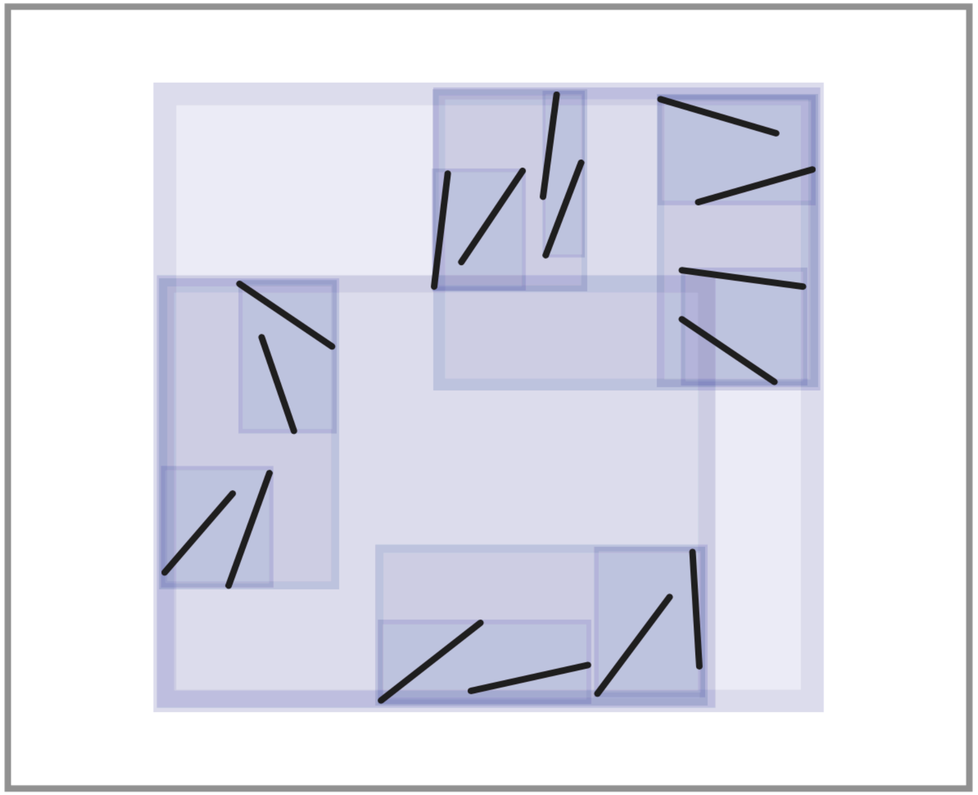
Example. Line segment ($\partial\Omega = A$)

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

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



Closest point queries

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

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

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

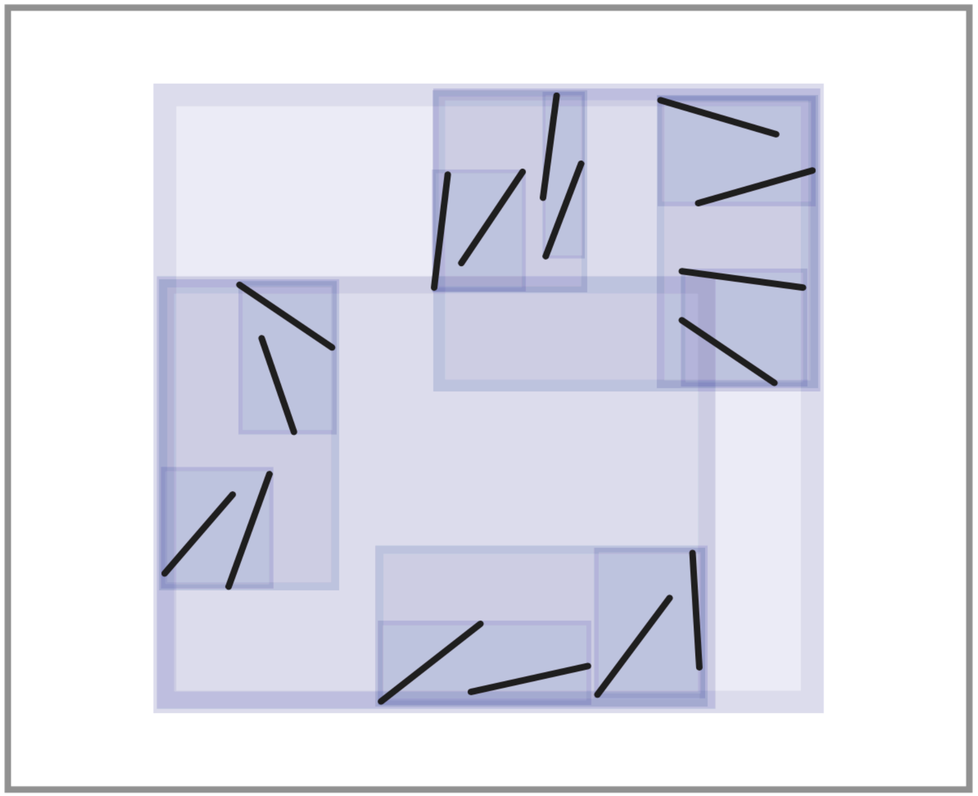
Example. Two line segment ($\partial\Omega = A \cup B$)

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

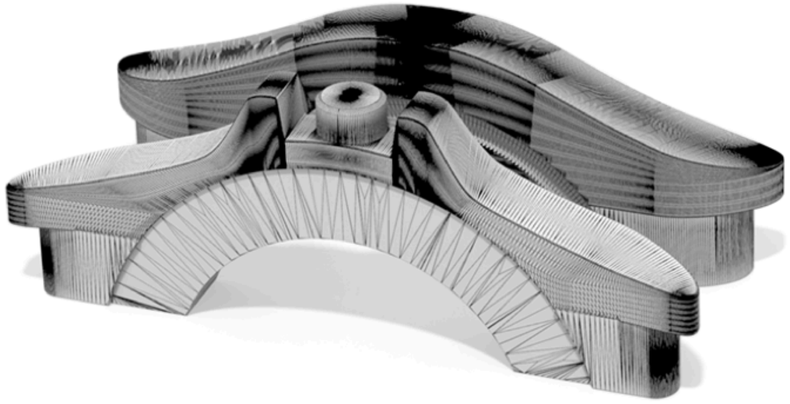
Example. Large number of line segments

Build *bounding volume hierarchy* (BVH)

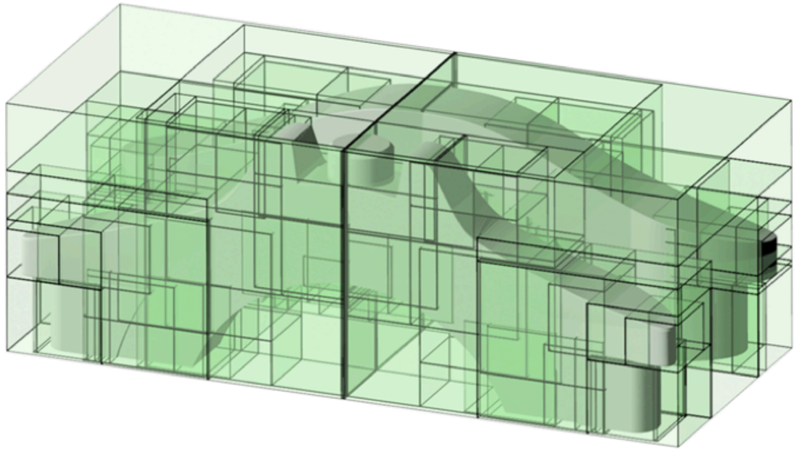
amortized cost of query is $O(\log n)$



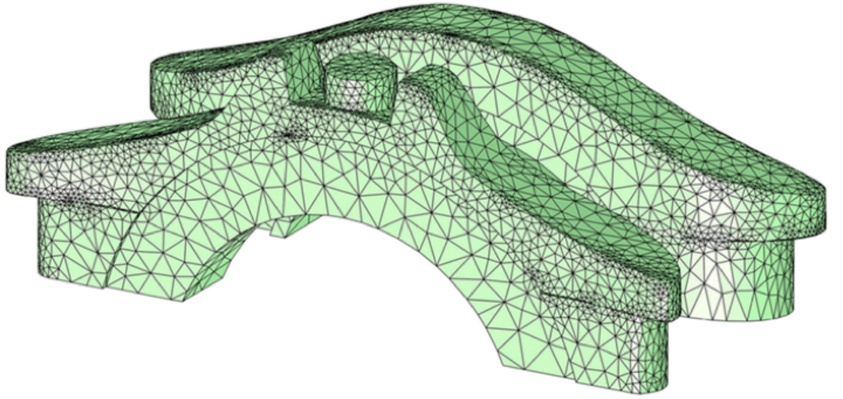
Benefits of Monte Carlo – Lightweight data-structure



input
boundary mesh



build BVH for WoS
few milliseconds



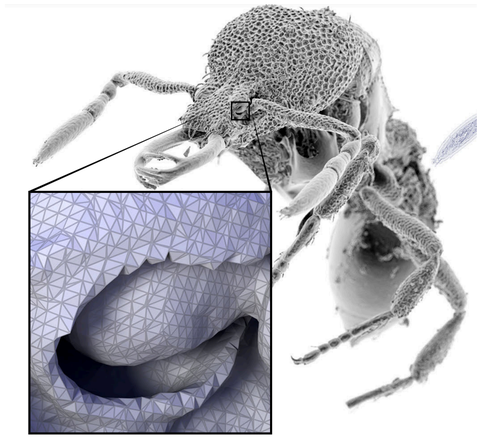
FEM mesh (FastTetWild)
1 hour 25 minutes

Geometric generality

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

explicit

mesh



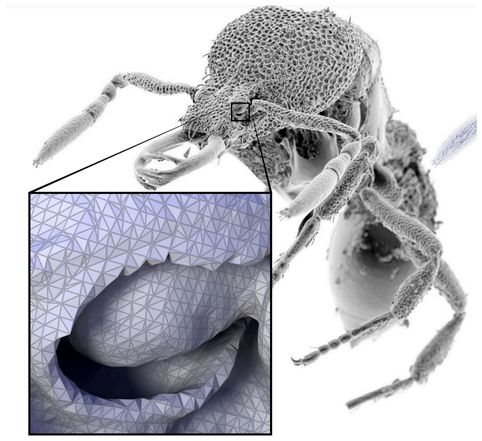
Geometric generality

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

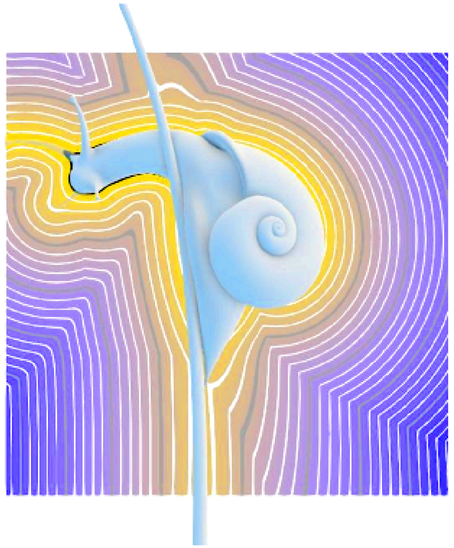
explicit

implicit

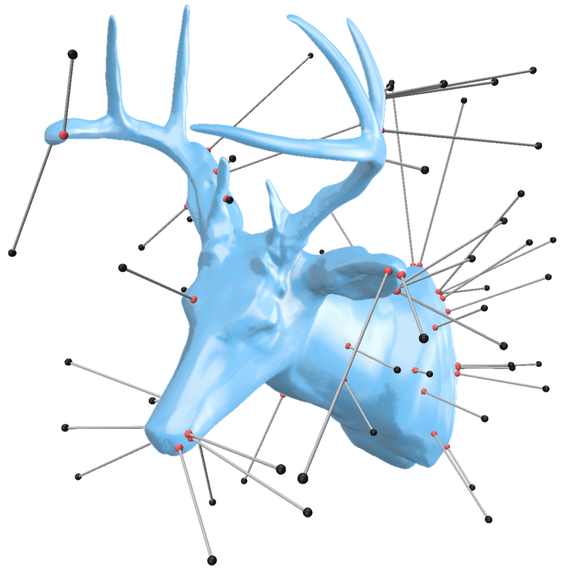
mesh



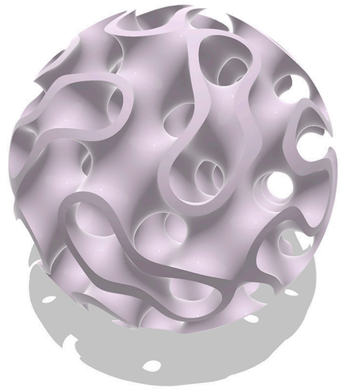
signed distance



neural



harmonic



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

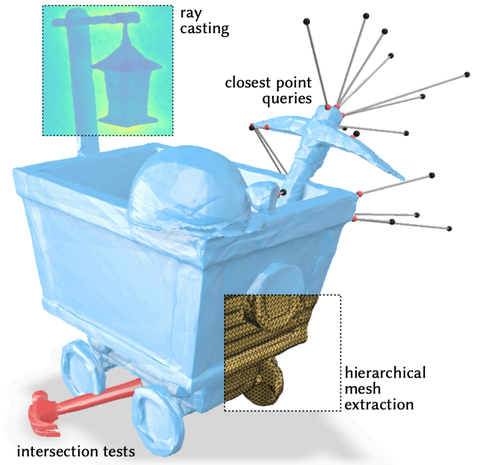


Fig. 1. Our method enables geometric queries on neural implicit surfaces, without relying on fitting a signed distance function. Several queries are shown here on a neural implicit occupancy function encoding a mine cart. These operations open up new explorations of deep implicit surfaces.

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

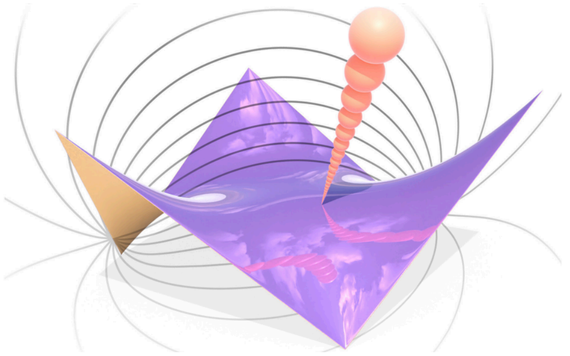
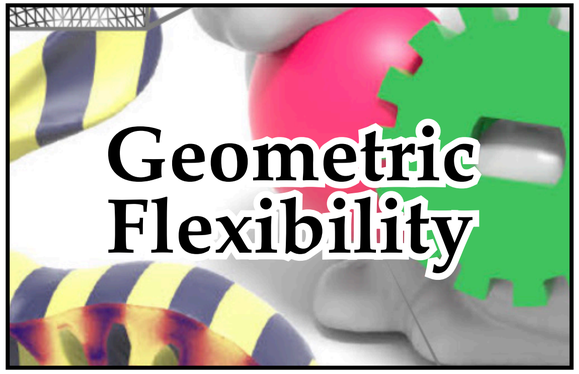
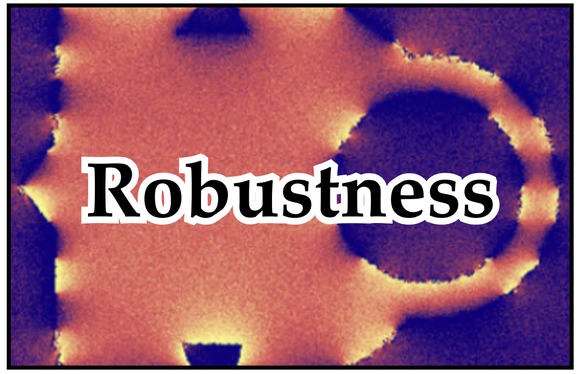


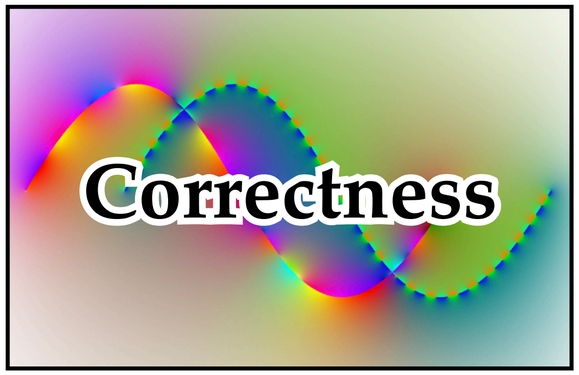
Fig. 1. We introduce a ray tracing algorithm for a novel class of surfaces defined by level sets of harmonic functions. Here for instance we directly visualize a nonplanar polygon which has no well-defined inside or outside—and hence cannot be represented by an ordinary implicit function or SDF. Isolines 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.

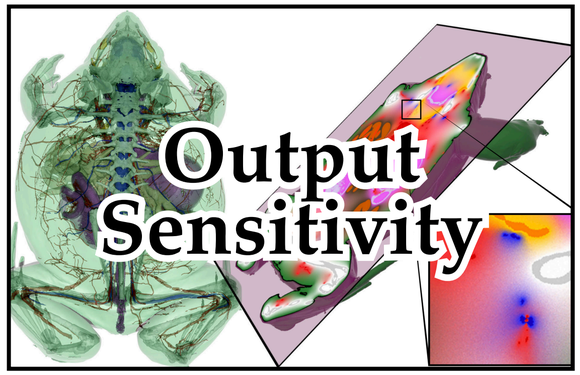
Benefits of Monte Carlo Methods

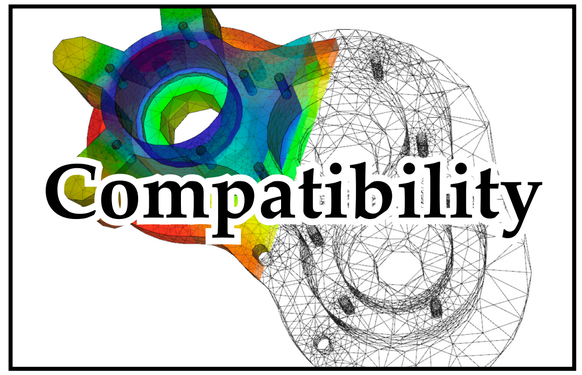
Monte Carlo PDE solvers provides many of the same benefits:









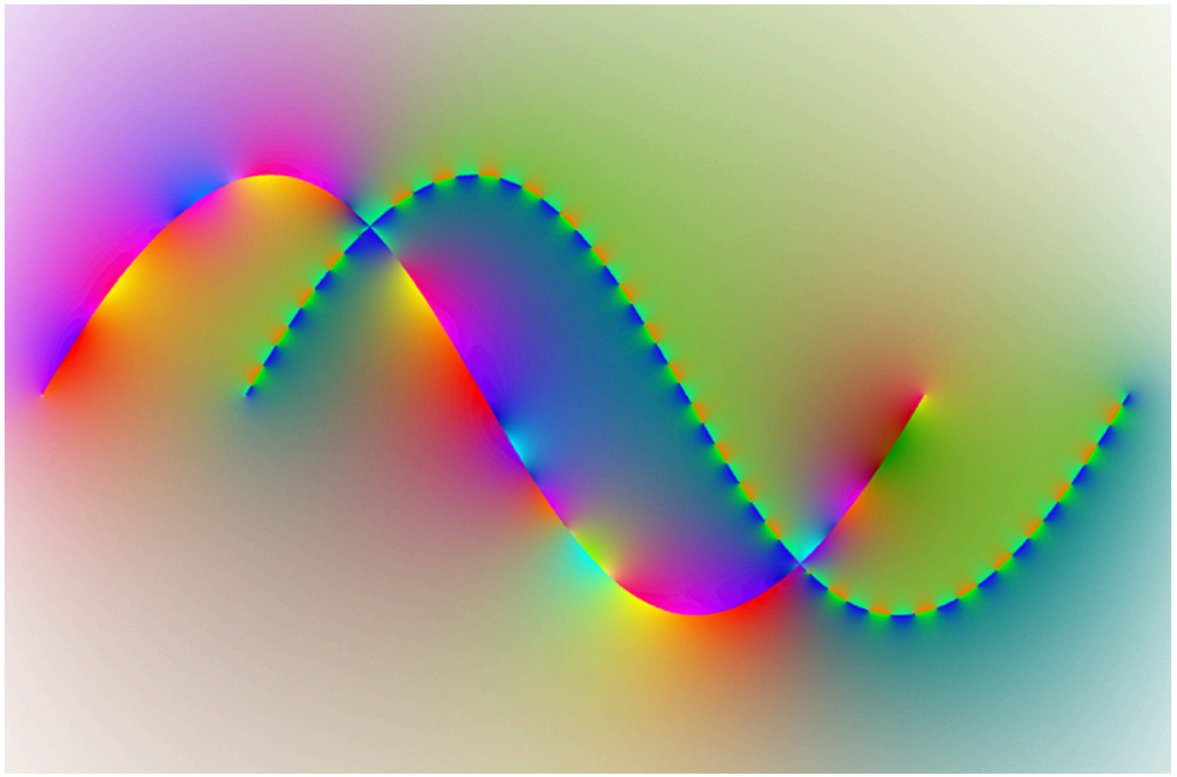


Benefits of Monte Carlo – Correctness

Rendering

[Wann Jensen 1995]

PDEs



[Sawhney & Crane 2020]

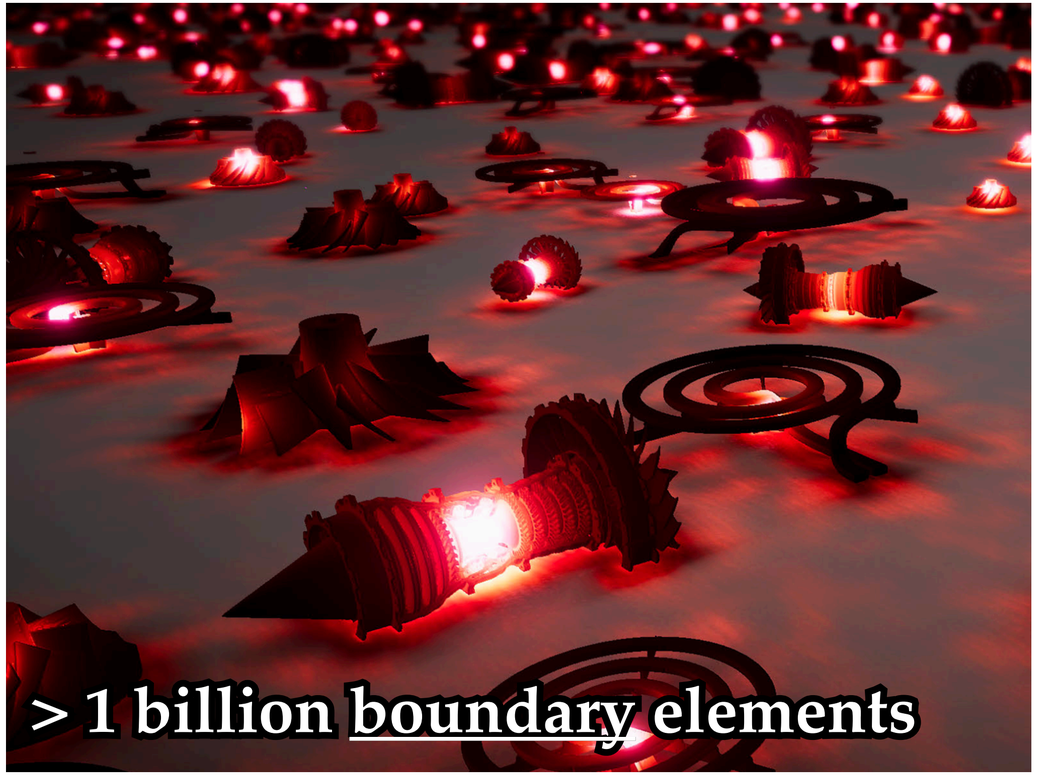
Key idea: as long as equation is well-posed, numerical solution *will* be correct

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

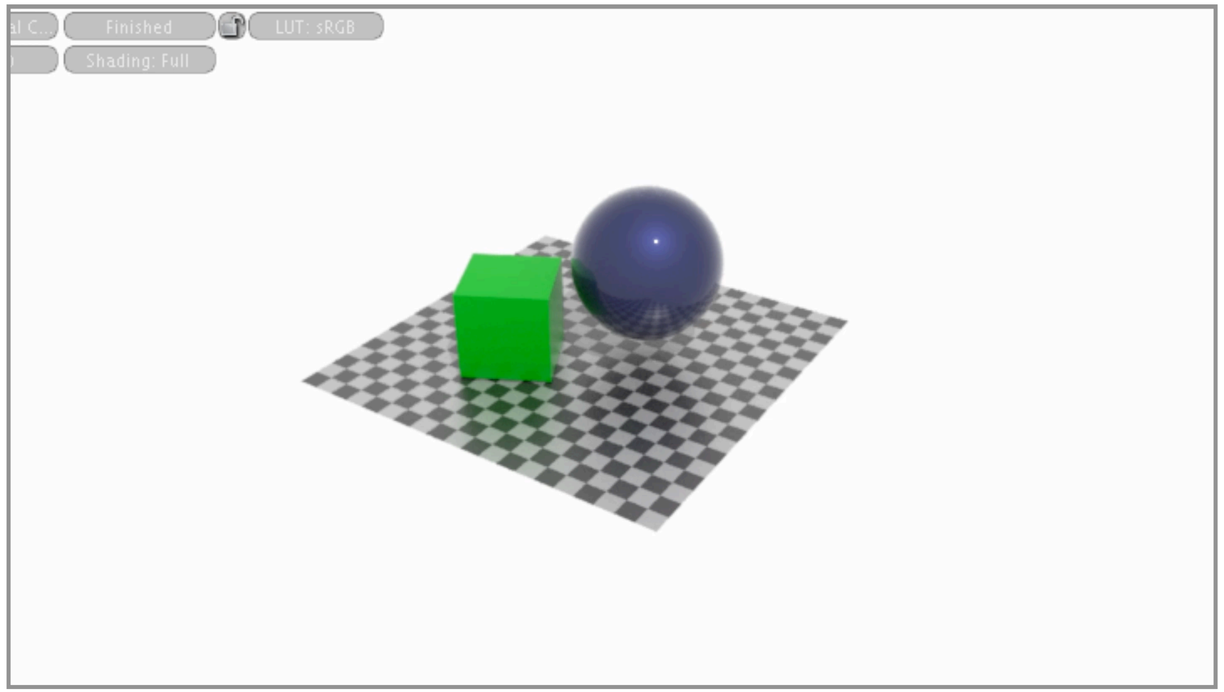
Rendering

PDEs

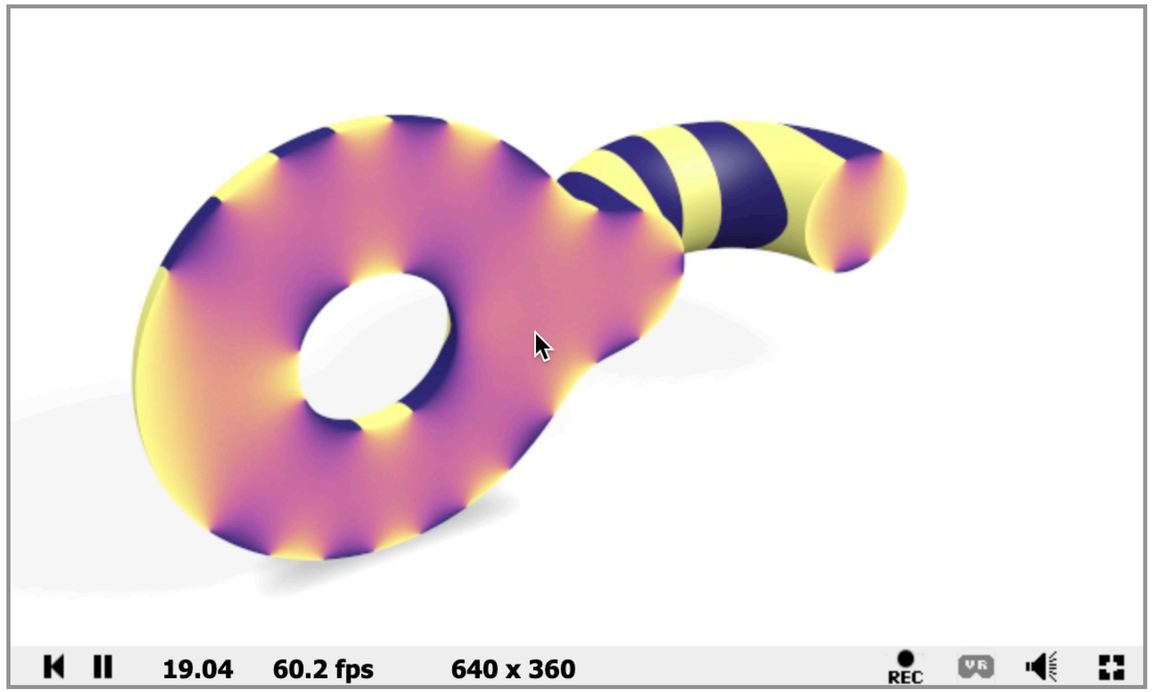
Key idea: just run on N processors, take average of N final estimates

Benefits of Monte Carlo – Progressive

Rendering



PDEs

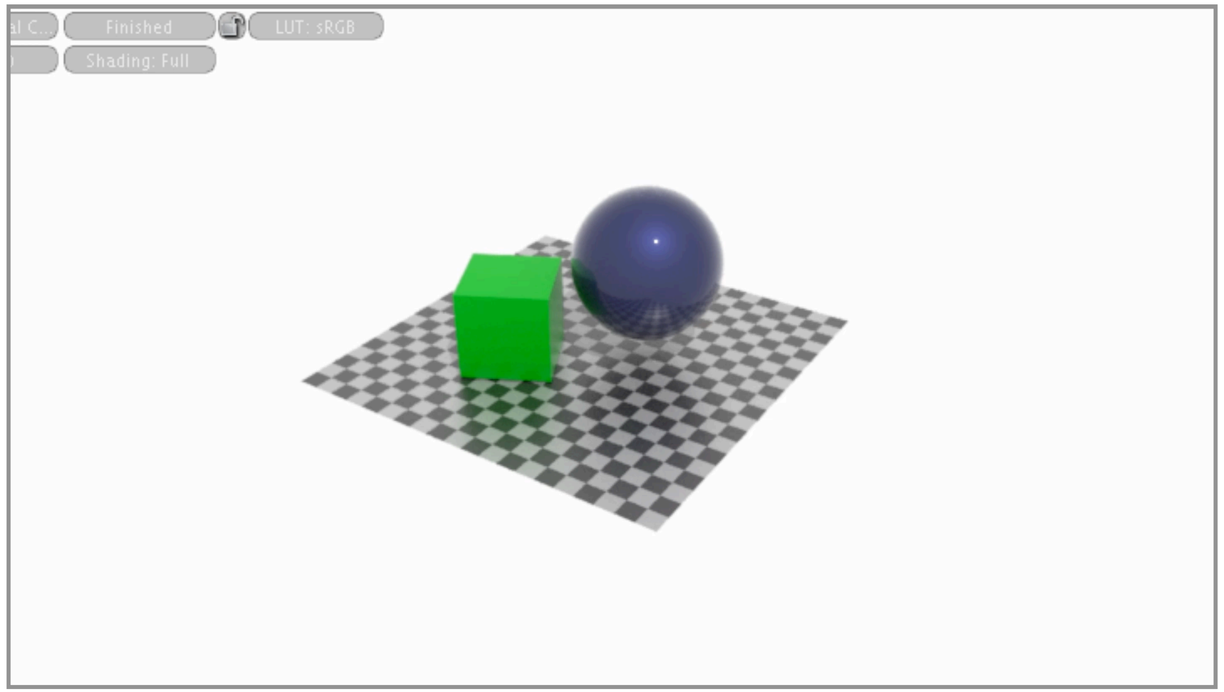


Credit: Ricky Reusser

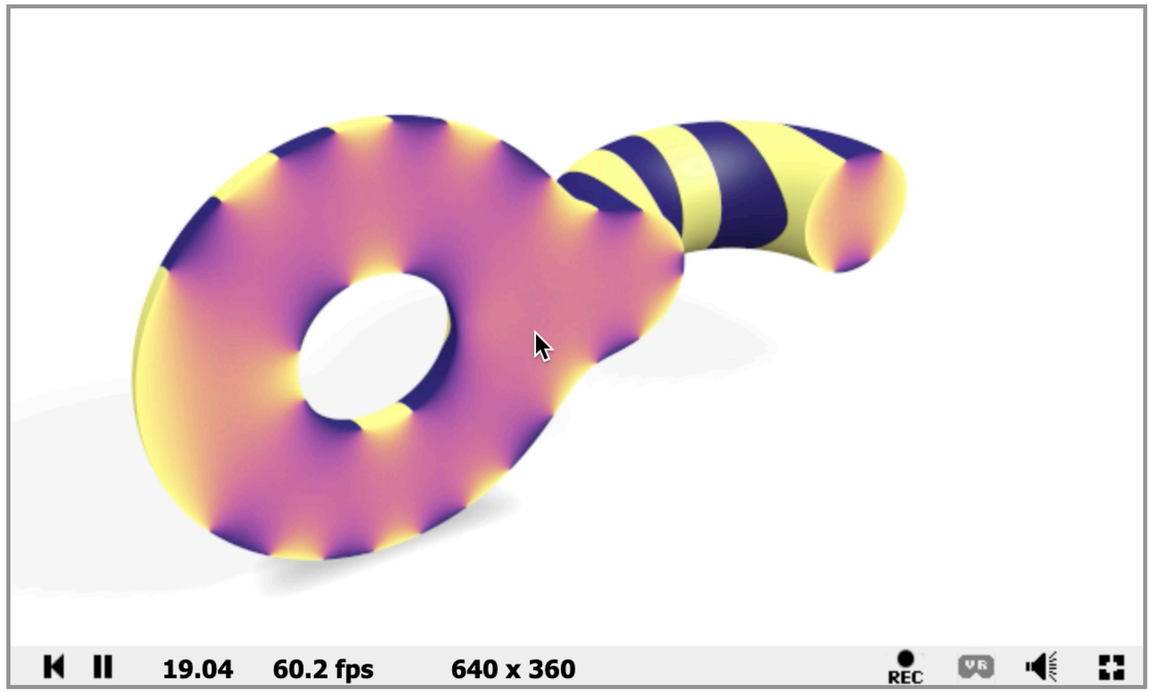
Key idea: fast-but-reliable “preview” enables instant exploration

Benefits of Monte Carlo – Progressive

Rendering



PDEs

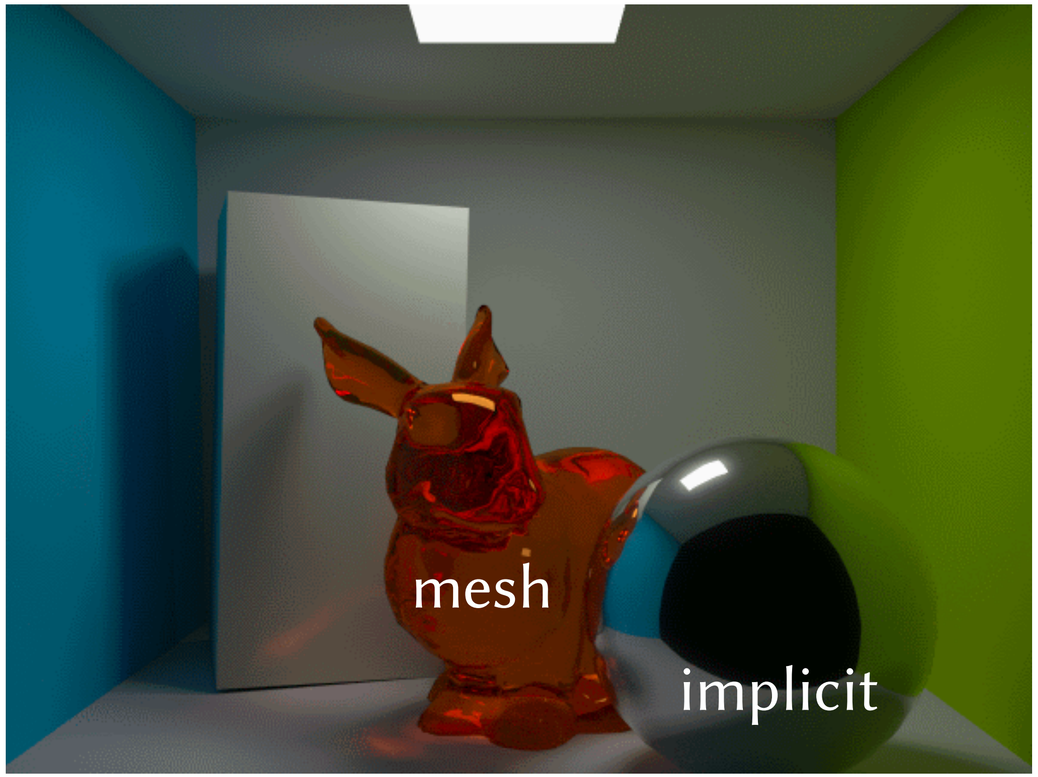


Credit: Ricky Reusser

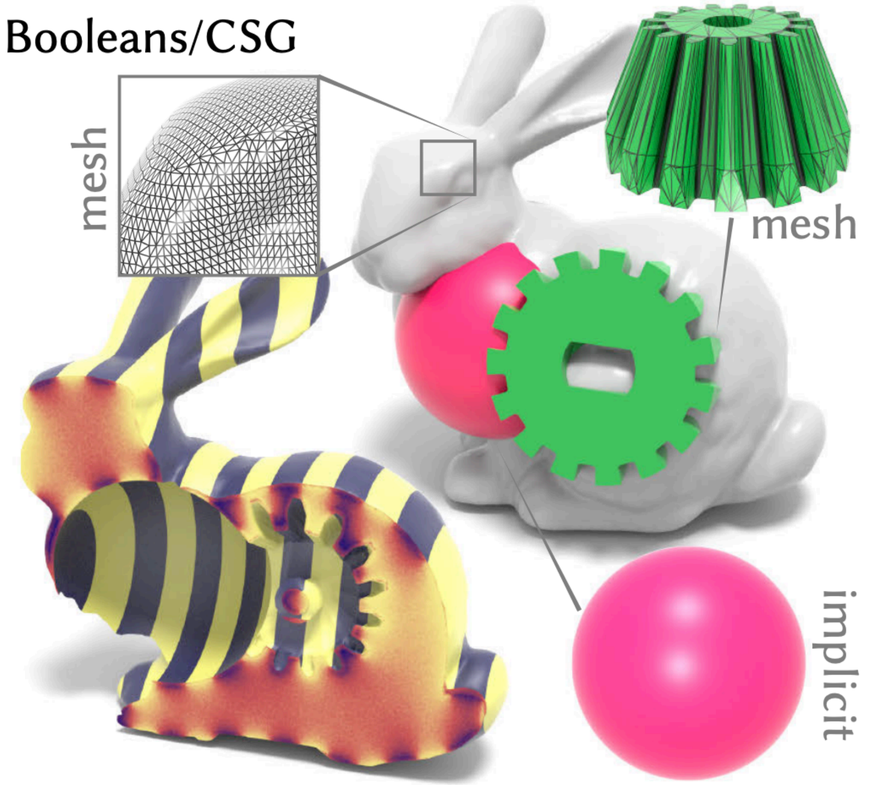
Key idea: fast-but-reliable “preview” enables instant exploration

Benefits of Monte Carlo – Geometric Generality

Rendering



PDEs

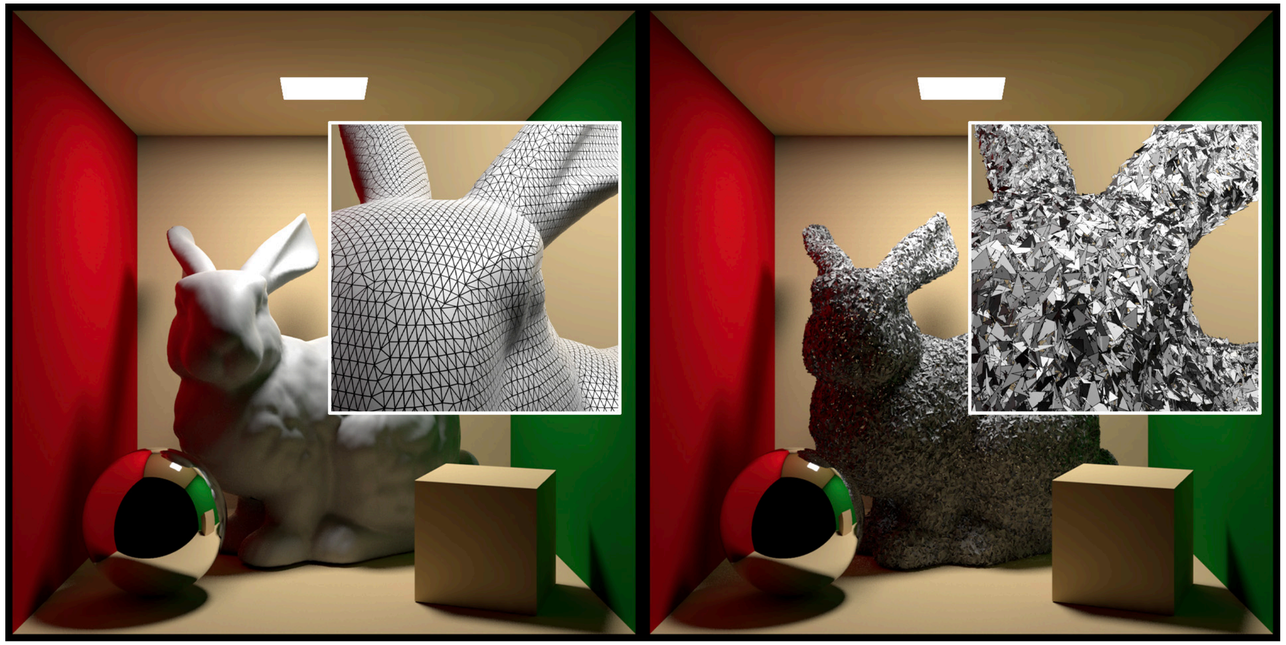


[Sawhney & Crane 2020]

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

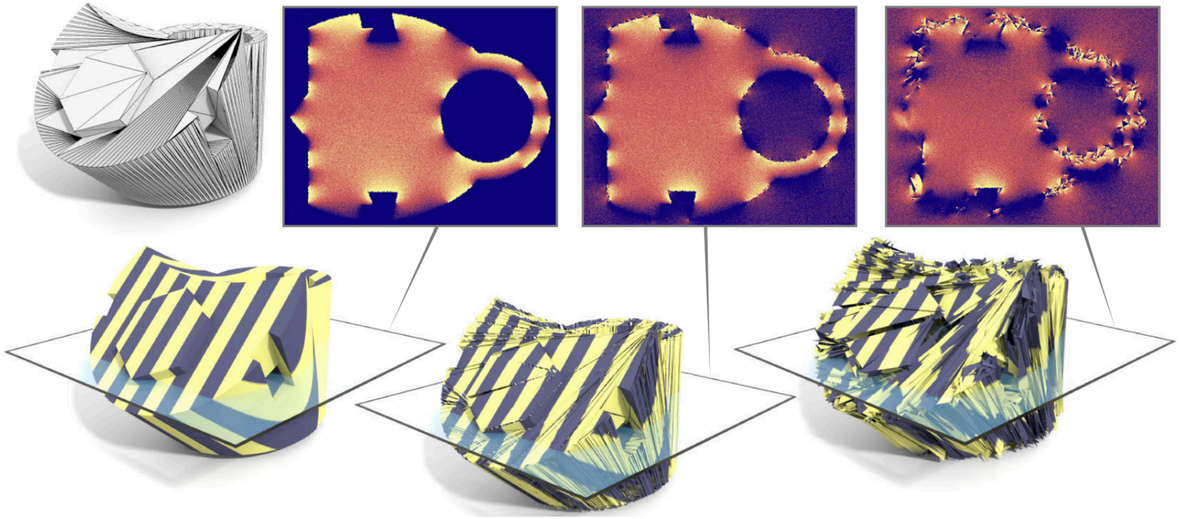
Benefits of Monte Carlo – Robustness

Rendering



Beautiful. Still beautiful.

PDEs



[Sawhney & Crane 2020]

Key idea: solution quality degrades *gracefully*

History of grid-free methods

History of grid-free methods

- **Theory & Algorithms**
  - Brownian motion related to heat & Laplace equations [Einstein 1905]

History of grid-free methods

- **Theory & Algorithms**
  - Brownian motion related to heat & Laplace equations [Einstein 1905]
  - *walk on spheres* [Muller 1956]

Mervin Muller

History of grid-free methods

- **Theory & Algorithms**
  - Brownian motion related to heat & Laplace equations [Einstein 1905]
  - *walk on spheres* [Muller 1956]

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 obtained without requiring any further restrictions on the boundary or the function defined on the boundary, in addition to those required for the existence and uniqueness of the solution to the Dirichlet problem. A detailed study is made for

History of grid-free methods

- **Theory & Algorithms**
  - Br
  - *wa*

n 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 obtained without requiring any further restrictions on the boundary or the function defined on the boundary, in addition to those required for the existence and uniqueness of the solution to the Dirichlet problem. A detailed study is made for

History of grid-free methods

- **Theory & Algorithms**
  - Brownian motion related to heat & Laplace equations [Einstein 1905]
  - *walk on spheres* [Muller 1956]

Mervin Muller

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]

Mervin Muller

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]

Mervin Muller

History of grid-free methods

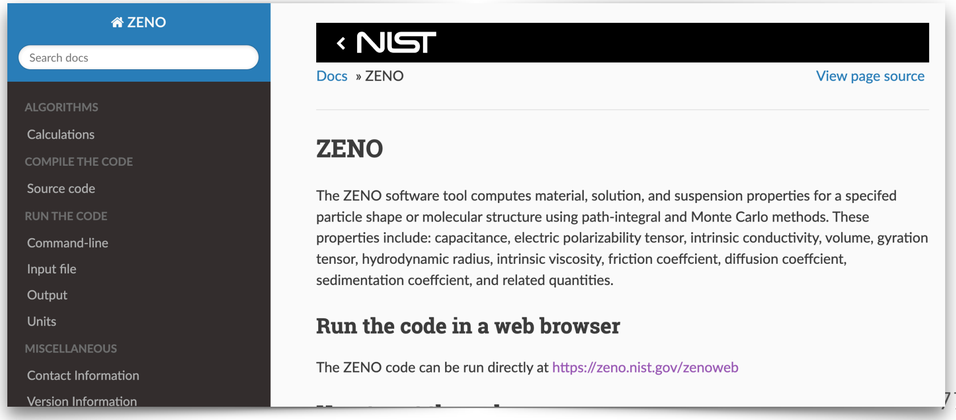
- **Theory & Algorithms**
  - Brownian motion related to heat & Laplace equations [Einstein 1905]
  - *walk on spheres* [Muller 1956]
  - a.k.a. *floating random walk* [Haji-Sheikh & Sparrow 1966]
  - complexity analysis [Binder & Braverman 2012]
  - extensions (Simonov, Sabelfeld, Mascagni, Deaconu, Booth, …)

Mervin Muller

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 [Coz & Iverson 1992]
  - porous media [Hwang et al 2000]
  - molecular dynamics [Mascagni & Simonov 2004]
  - *…not much else!*

Mervin Muller



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 [Coz & Iverson 1992]
  - porous media [Hwang et al 2000]
  - molecular dynamics [Mascagni & Simonov 2004]
  - *…not much else!*

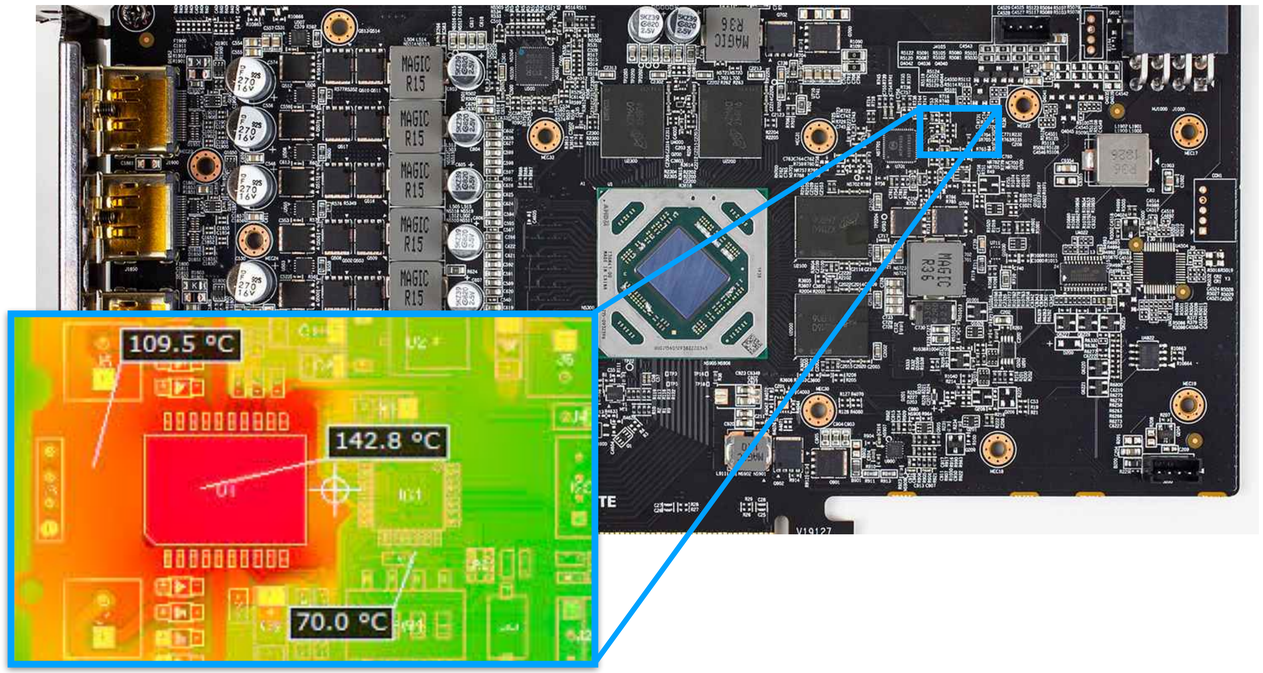
Mervin Muller

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

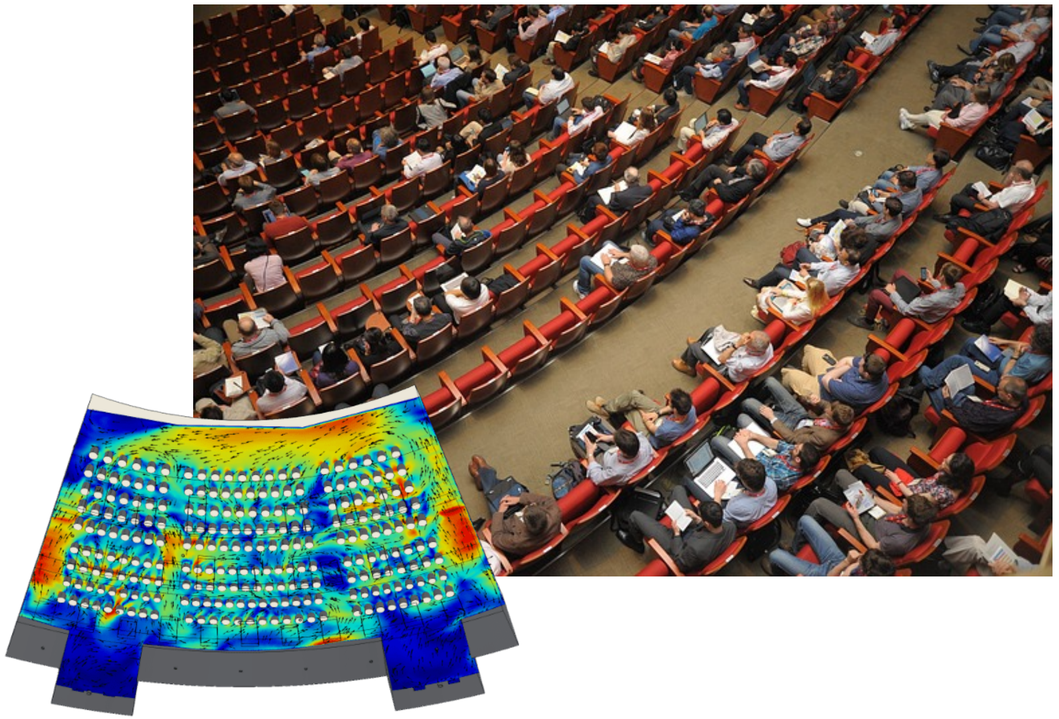
PART 2:
WoS for Poisson Equation

Motivation for source terms

thermal analysis of PCBs



thermal management in building design



Roadmap for solving Poisson PDE

Roadmap for solving Poisson PDE

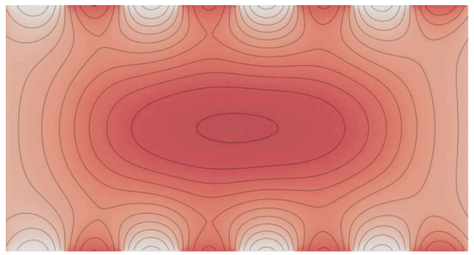
differential form

Poisson PDE

Find u such that

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

$u(x)$



$f(x)$

$g(x)$

Roadmap for solving Poisson PDE

differential form

Poisson PDE

Find u such that

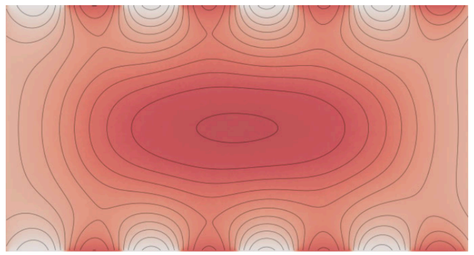
$$\Delta u = f \quad \text{on } \Omega$$
$$u = g \quad \text{on } \partial\Omega$$

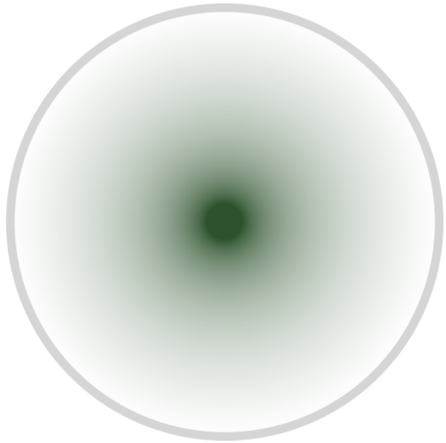
integral representation

Green's integral

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

$u(x)$





$f(x)$

$g(x)$

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

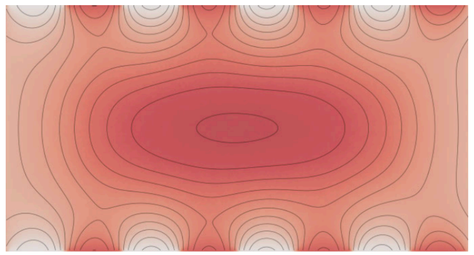
$$\int_{B(x)} G(x,z) f(z)\, dz$$

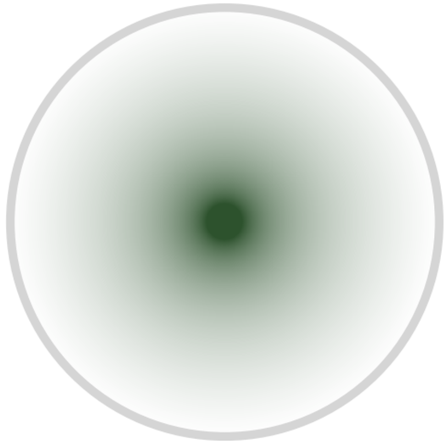
Monte Carlo estimator

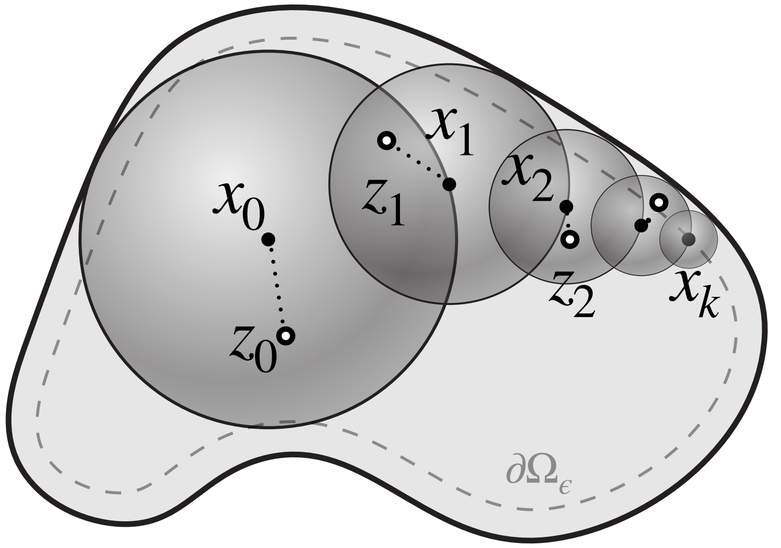
Walk on Spheres+

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

$u(x)$





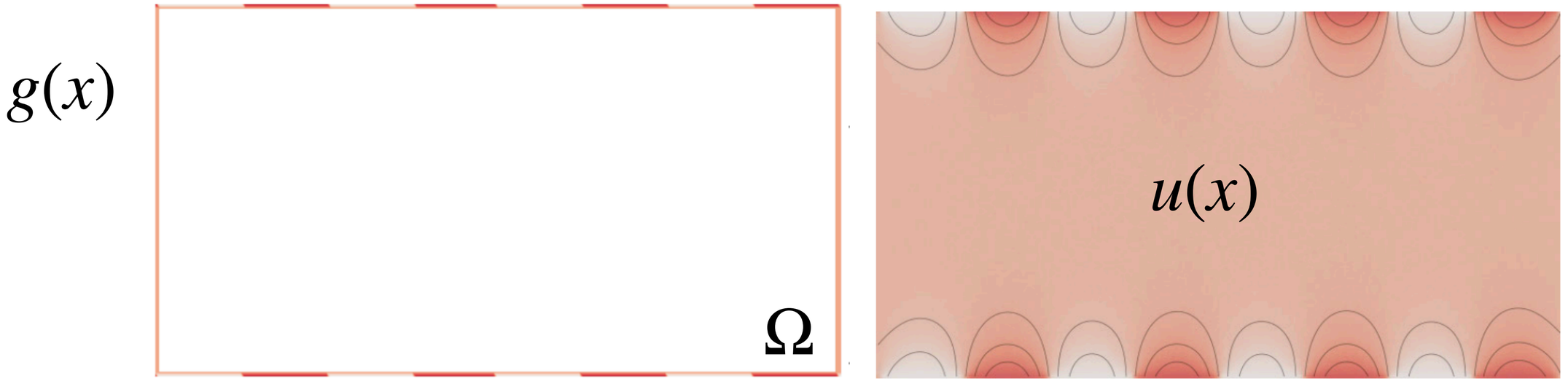


$f(x)$

$g(x)$

Dirichlet boundary condition

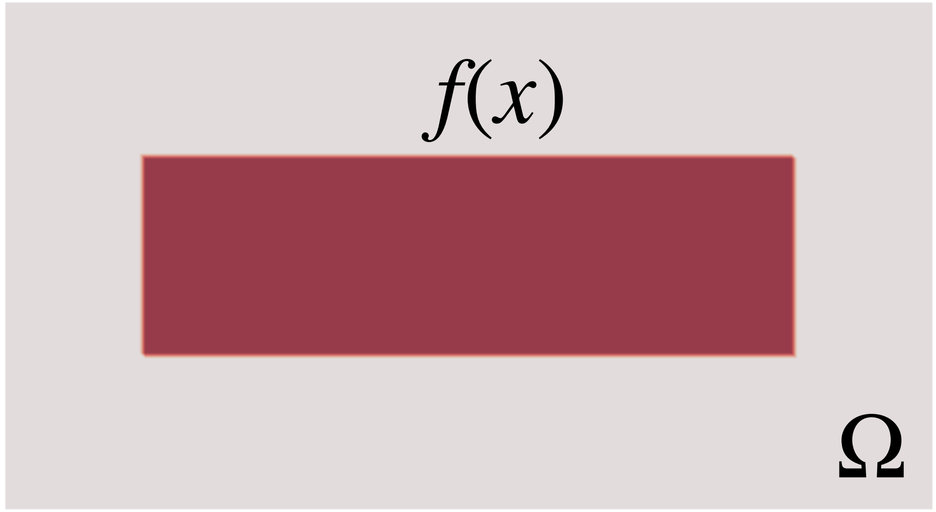
$$\Delta u = 0 \quad \text{on } \Omega$$
$$u = g \quad \text{on } \partial\Omega$$

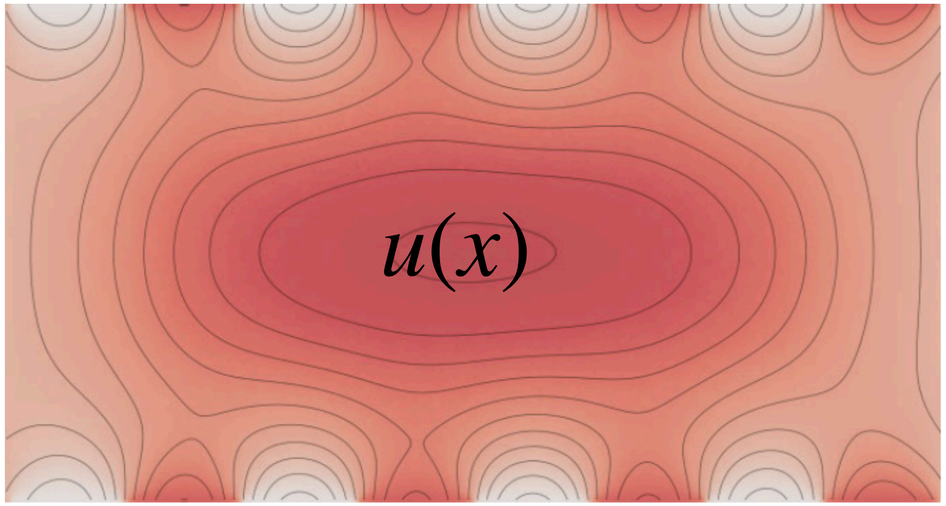


Intuition: temperature prescribed on the boundary

Poisson equation

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





Intuition: adds additional background temperature

Green's function

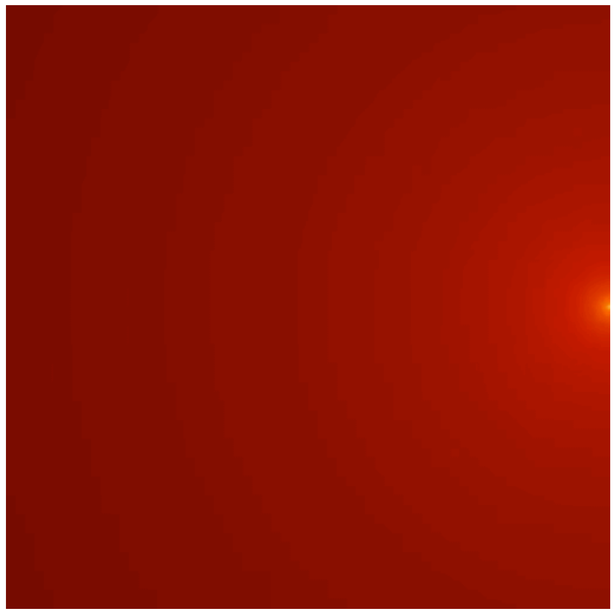
$$\Delta G(x,z) = \delta_z(x)$$

heat injected at single point

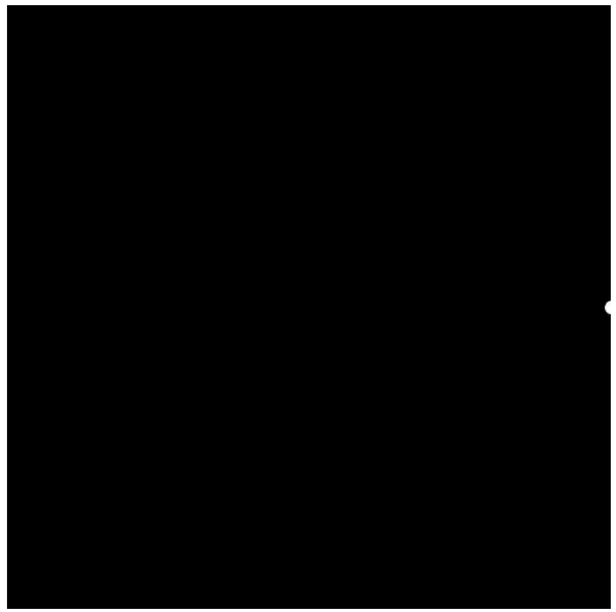
free-space Green's function

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

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



$G(x,z)$



$\delta_z(x)$

Green's function

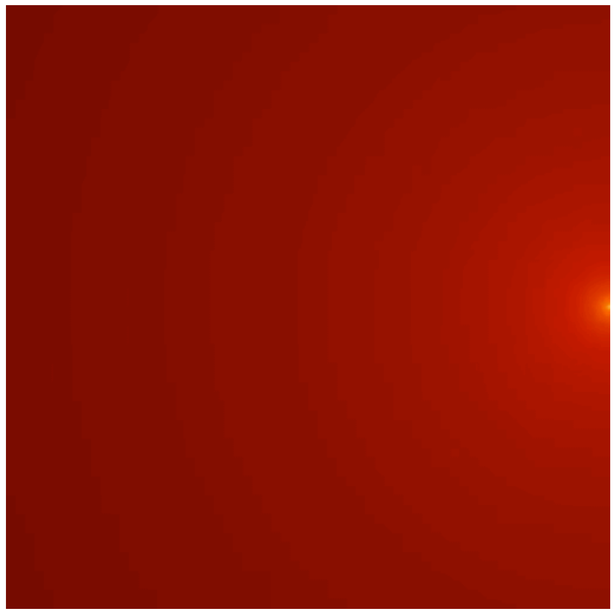
$$\Delta G(x,z) = \delta_z(x)$$

heat injected at single point

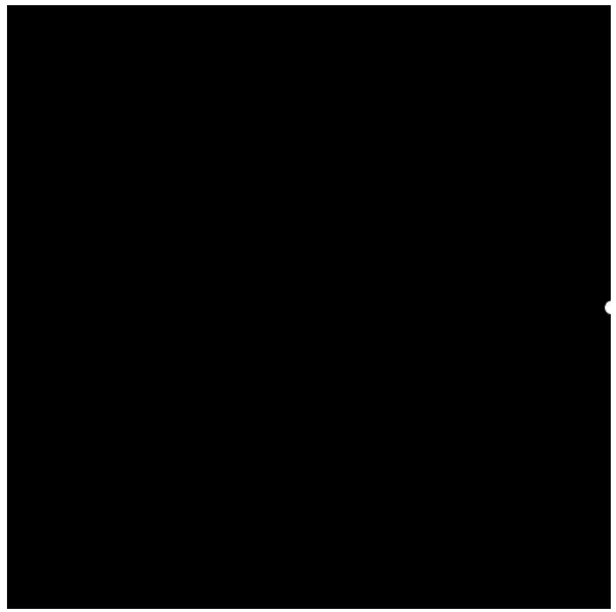
free-space Green's function

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

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



$G(x,z)$



$\delta_z(x)$

Green's function

Green's function on a ball

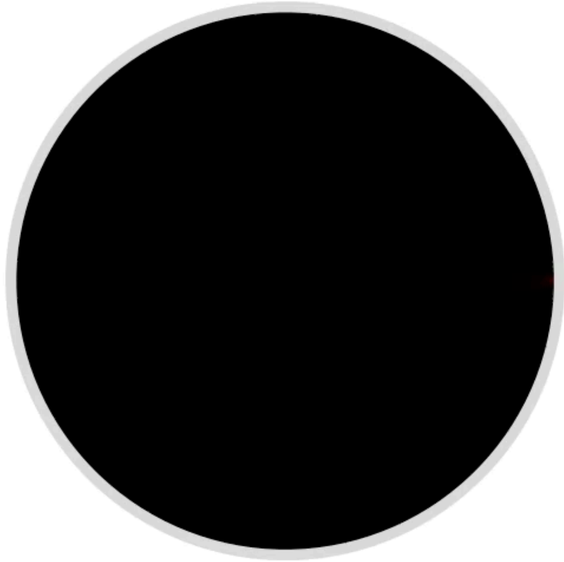
$$G_B^{2D}(x,z) = \frac{1}{2\pi} \ln\left(\frac{R^2 - \langle x,z\rangle}{R\,|z-x|}\right)$$

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

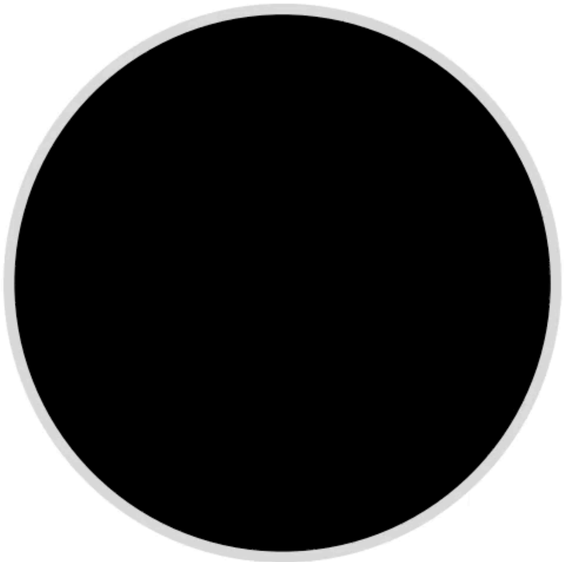
$$\Delta G_B(x,z) = \delta_z(x) \quad \text{on } B$$

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

zero temperature at boundary



$G_B(x,z)$



$\delta_z(x)$

Green's function

Green's function on a ball

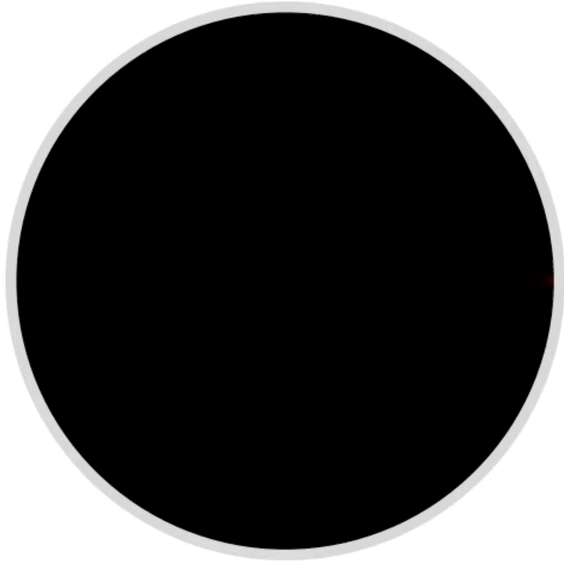
$$G_B^{2D}(x,z) = \frac{1}{2\pi} \ln\left(\frac{R^2 - \langle x,z\rangle}{R\,|z-x|}\right)$$

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

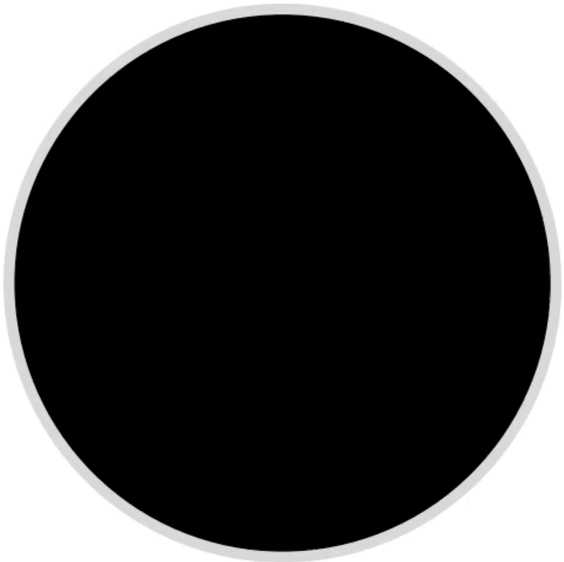
$$\Delta G_B(x,z) = \delta_z(x) \quad \text{on } B$$

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

zero temperature at boundary



$G_B(x,z)$



$\delta_z(x)$

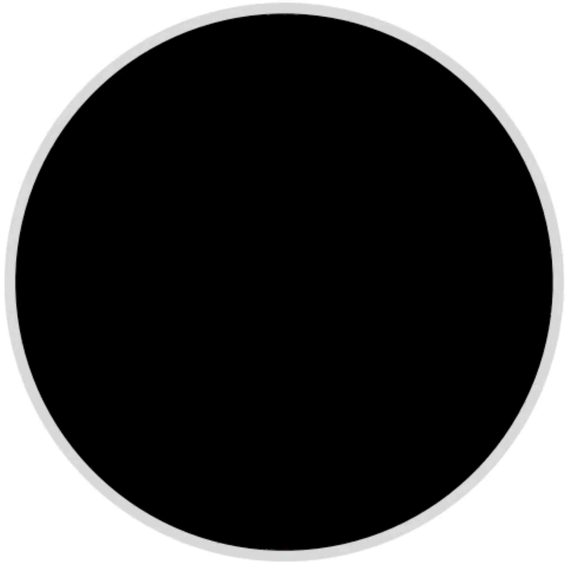
Sources are additive

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

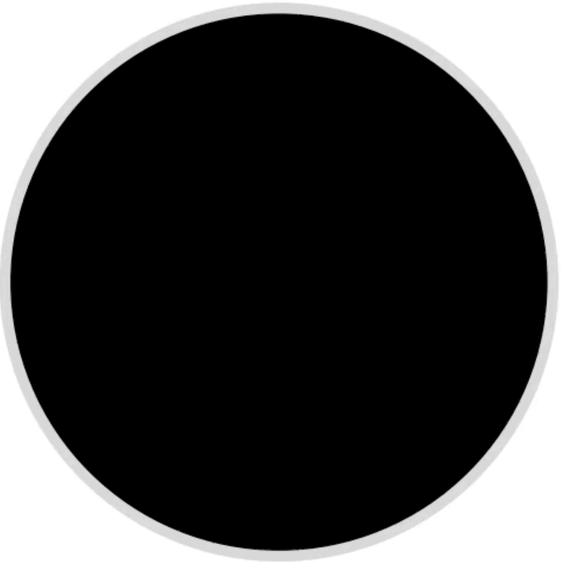
$$\Delta w(x) = \delta_{z_1}(x) + \delta_{z_2}(x) \quad \text{on } B$$
$$w(x) = 0 \quad \text{on } \partial B$$

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

$w(x)$



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



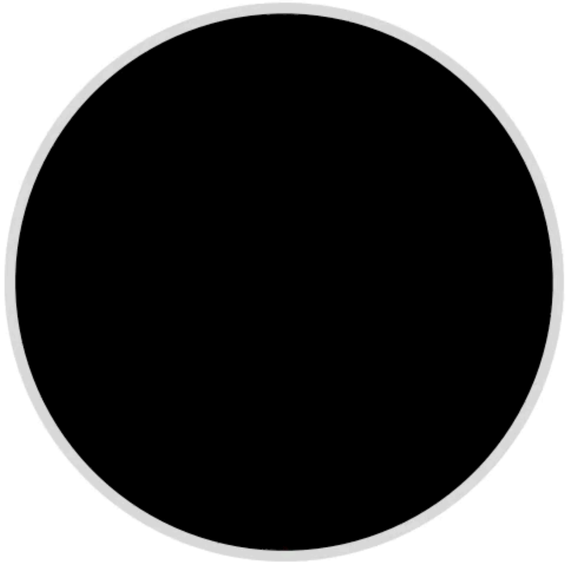
Sources are additive

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

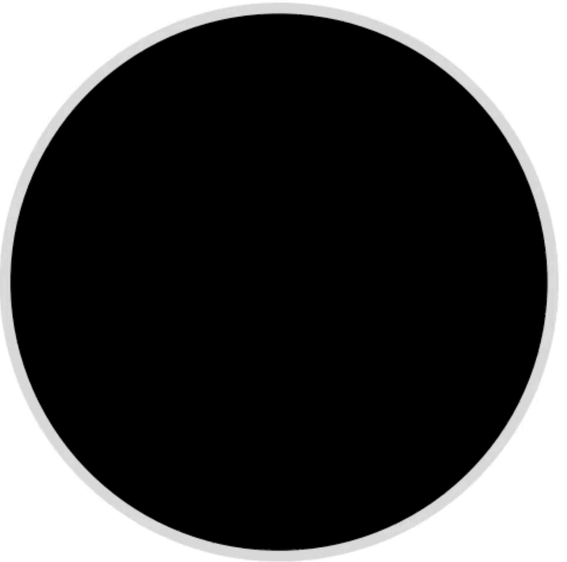
$$\Delta w(x) = \delta_{z_1}(x) + \delta_{z_2}(x) \quad \text{on } B$$
$$w(x) = 0 \quad \text{on } \partial B$$

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

$w(x)$



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



Green's integral on a ball

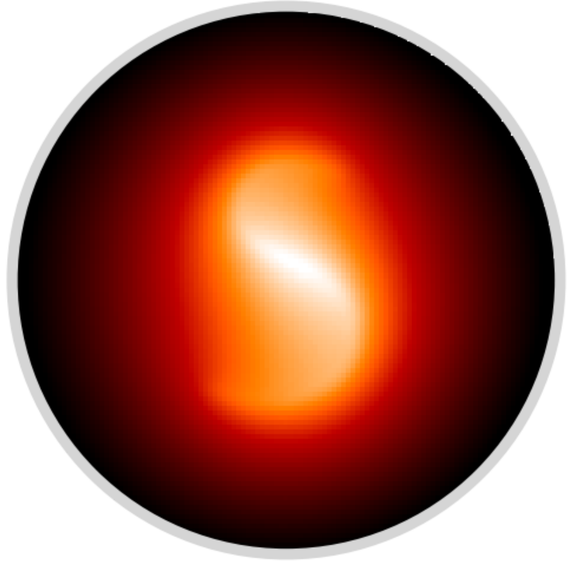
By convolving the Green's integral with a source term we obtain the source contribution over the ball.

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

Green's integral

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

$w(x)$



$f(z)$

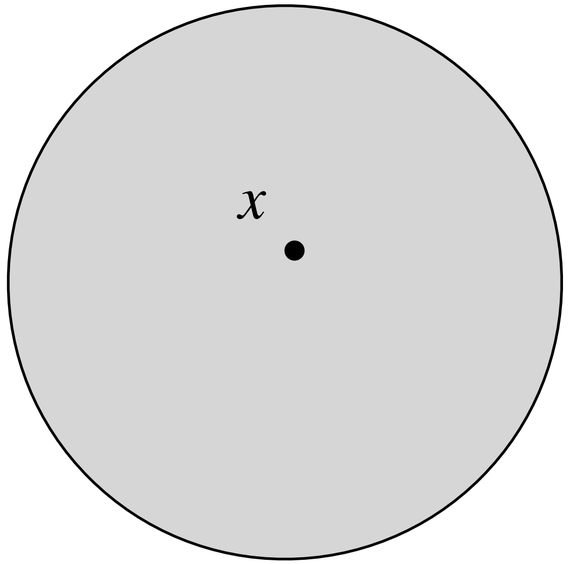
Generalized mean value integral

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

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



Generalized mean value integral

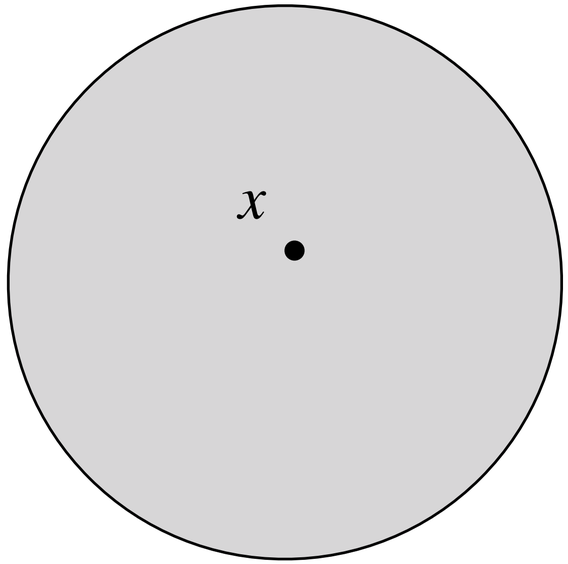
The mean value integral is generalized by considering the source contribution within the ball centered at x.

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

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



Generalized mean value integral

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

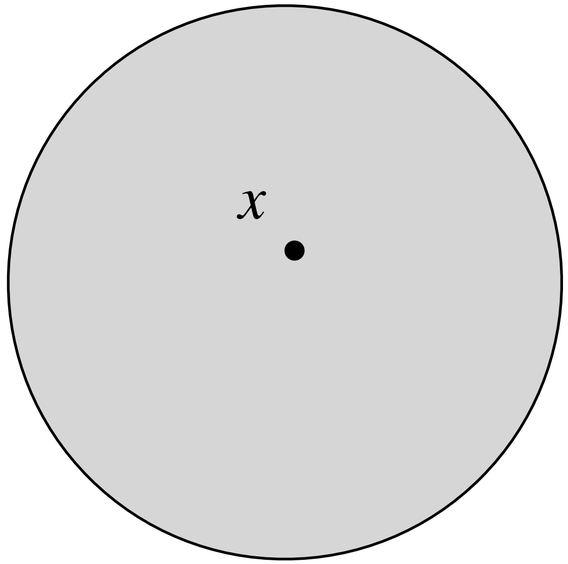
$$\Delta u = f \quad \text{on } \Omega$$
$$u = g \quad \text{on } \partial\Omega$$

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 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

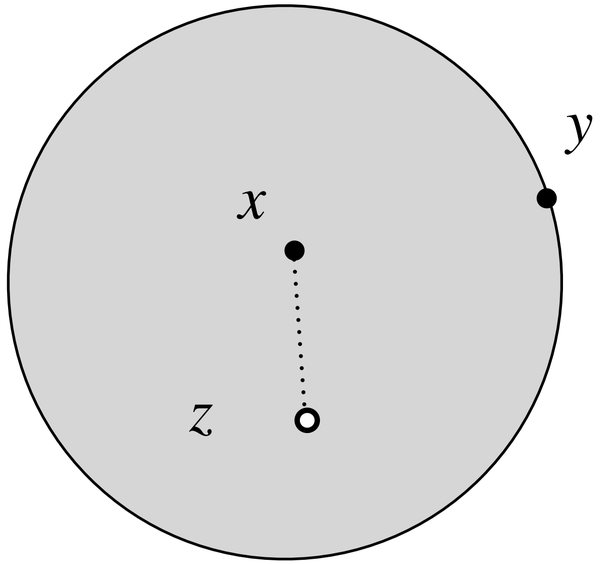
$$u(x) = \frac{1}{|\partial B(x)|}\int_{\partial B(x)} u(y)\, dy + \int_{B(x)} G(x,z) f(z)\, dz$$

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

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

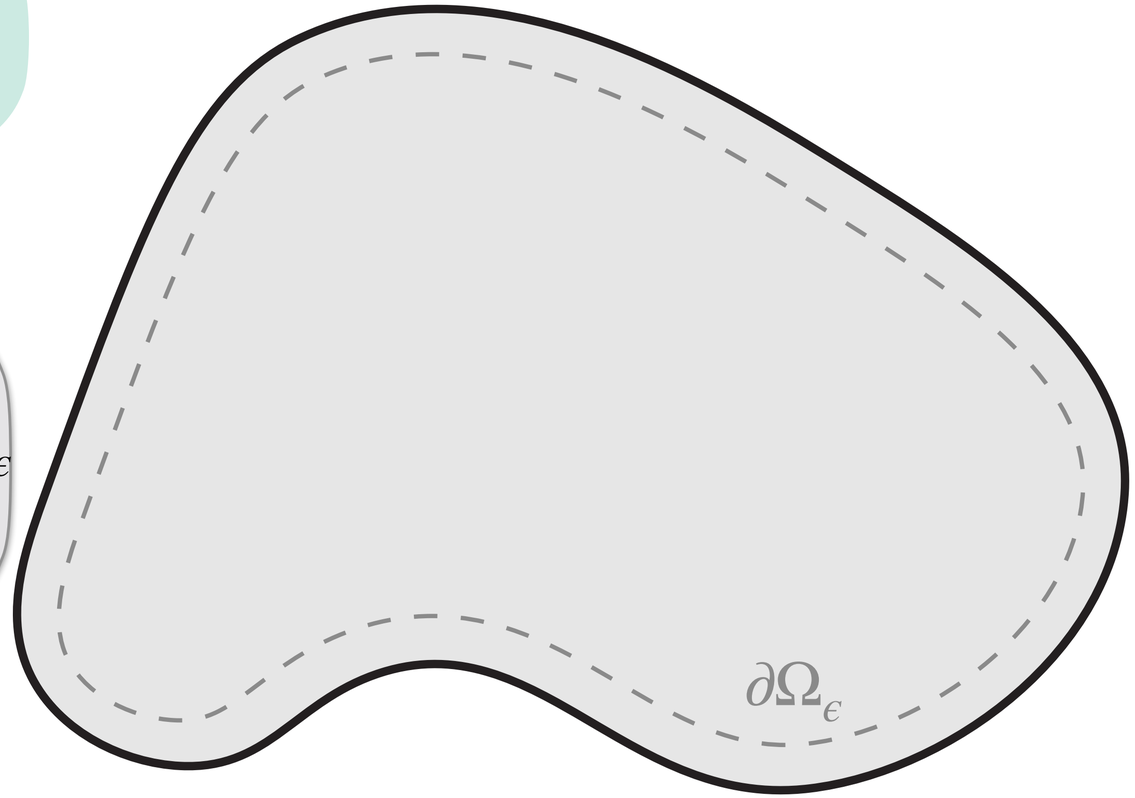
boundary

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

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

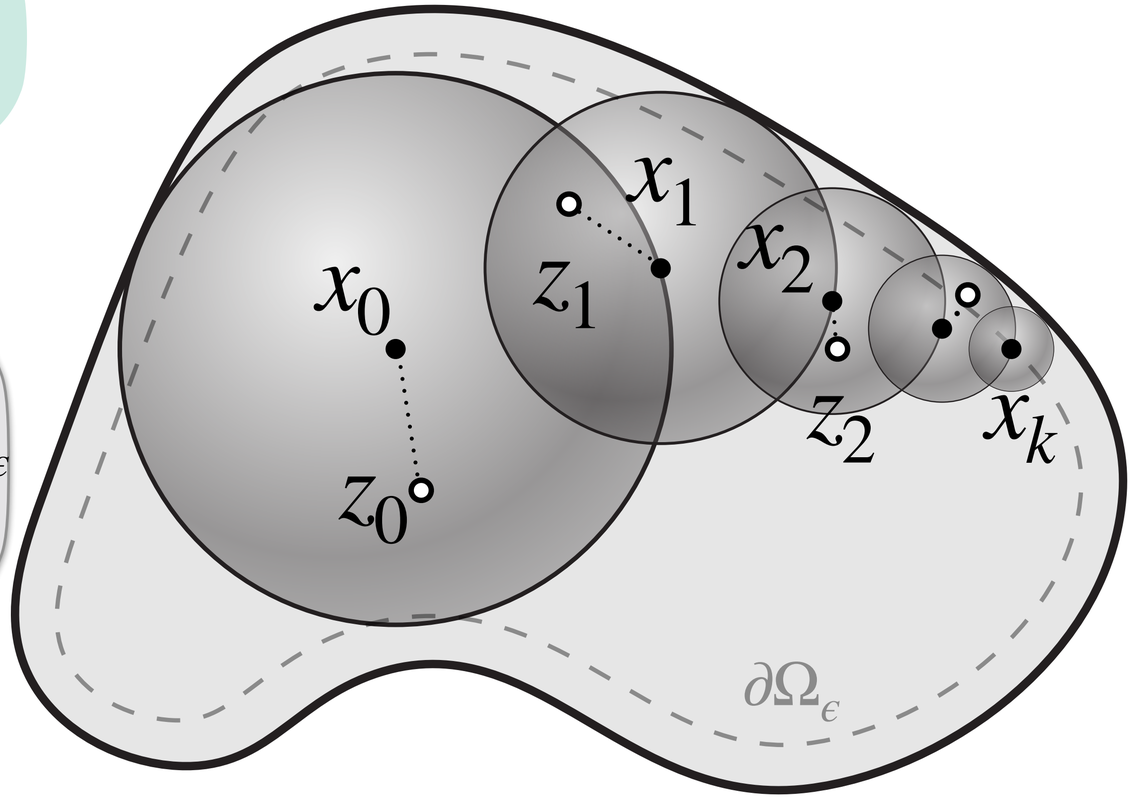
$$u(x) = \frac{1}{|\,\partial B(x)\,|}\int_{\partial B(x)} u(y)\; dy + \int_{B(x)} G(x,z) f(z)\; dz$$

boundary source

walk on spheres estimator

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

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



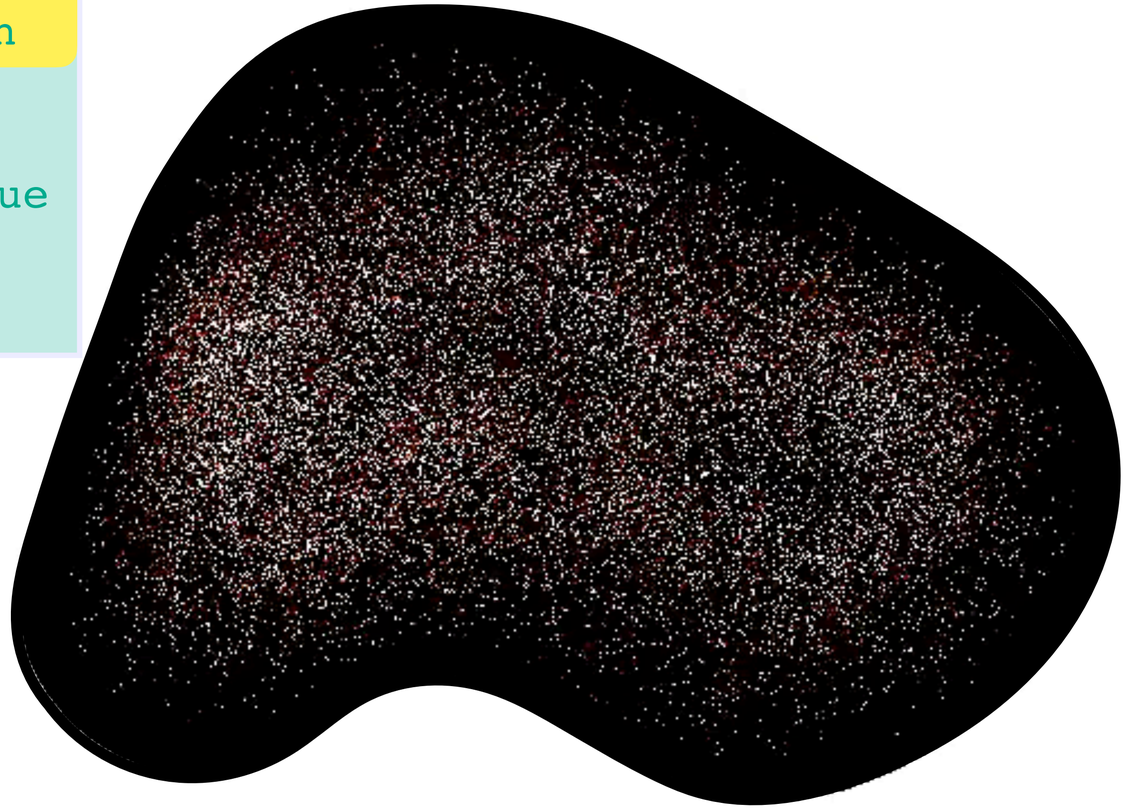
Walk on spheres with source term

```
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
```

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

walk on spheres estimator

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



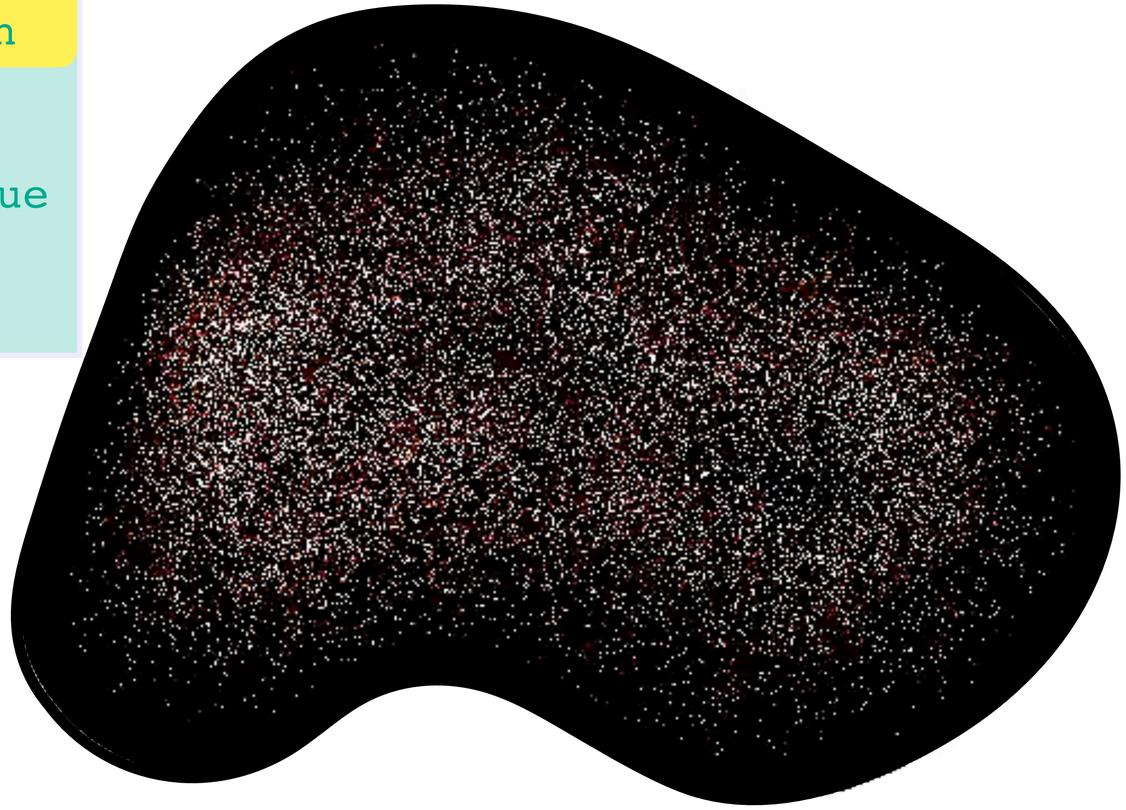
Walk on spheres with source term

```
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
```

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

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

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

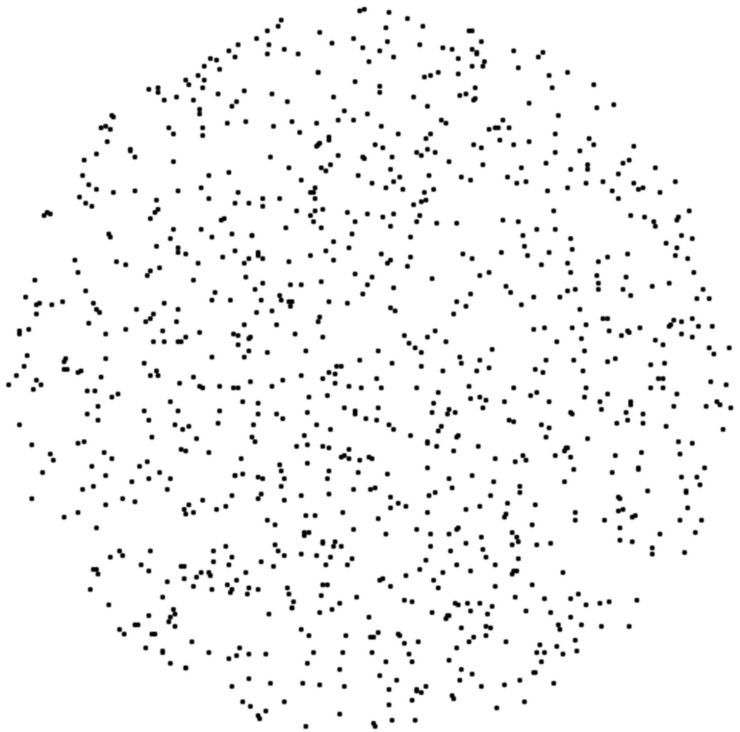
$$\hat{I}_{\text{uniform}} = \frac{|B(x)|}{n} \sum_{i=1}^{n} G(x, z_i) f(z_i)$$

Uniform 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

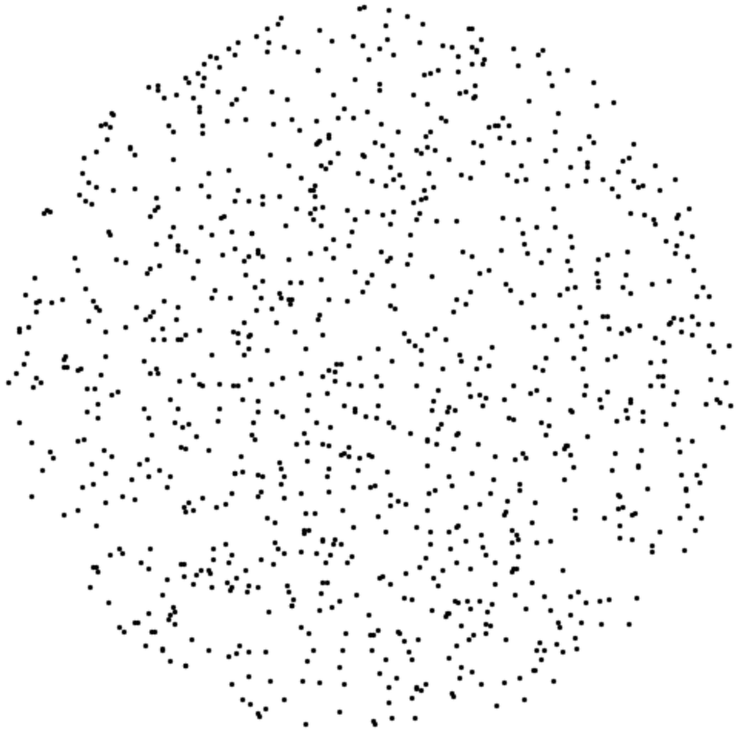


Uniform sampling limitations

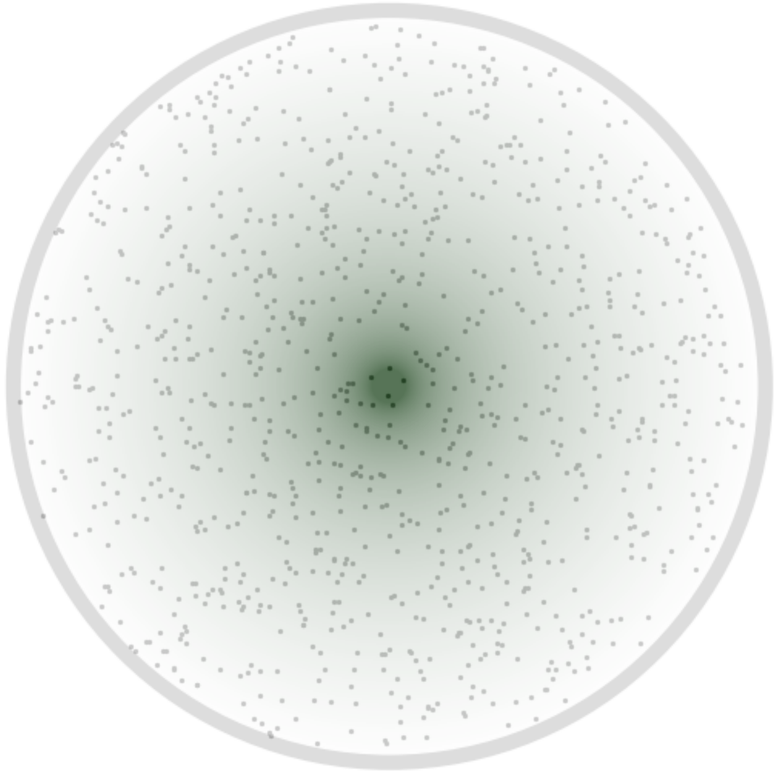
$$I = \int_{B(x)} G(x, z) f(z)\ dz$$

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

uniform samples



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



Uniform sampling limitations

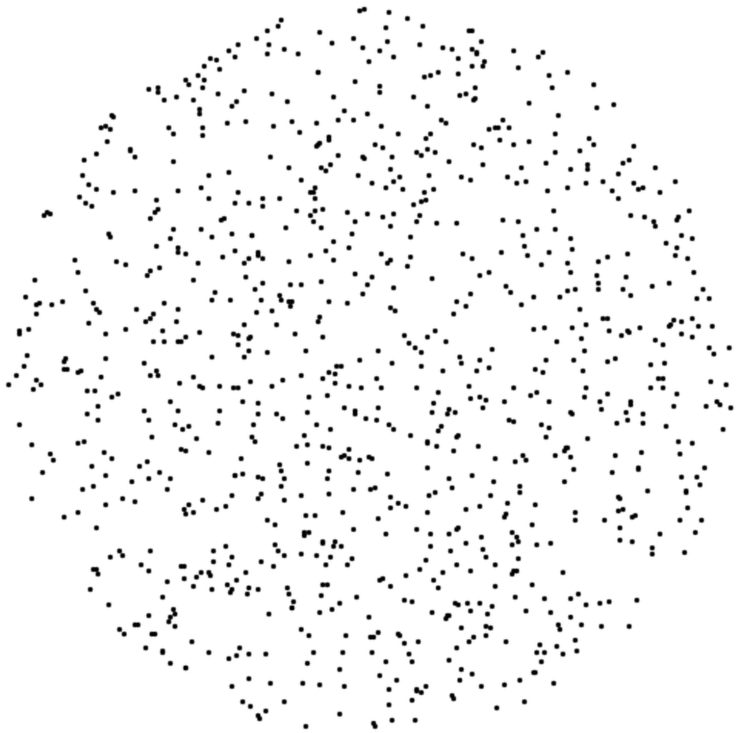
$$I = \int_{B(x)} G(x, z) f(z)\ dz$$

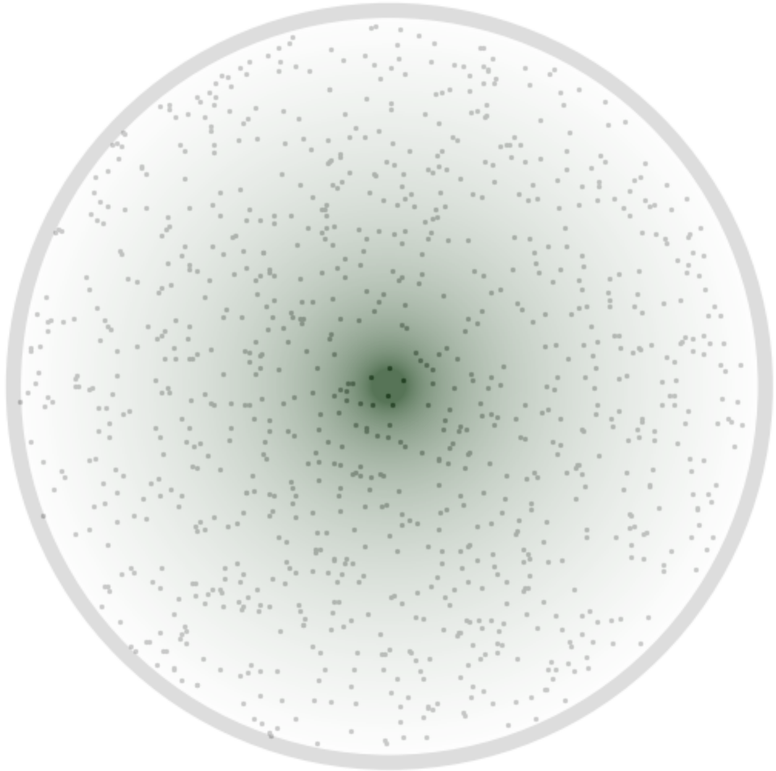
$$\hat{I}_{\text{uniform}} = \frac{|B(x)|}{n} \sum_{i=1}^{n} G(x, z_i) f(z_i)$$

uniform samples

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

source function $f(z)$





Uniform sampling limitations

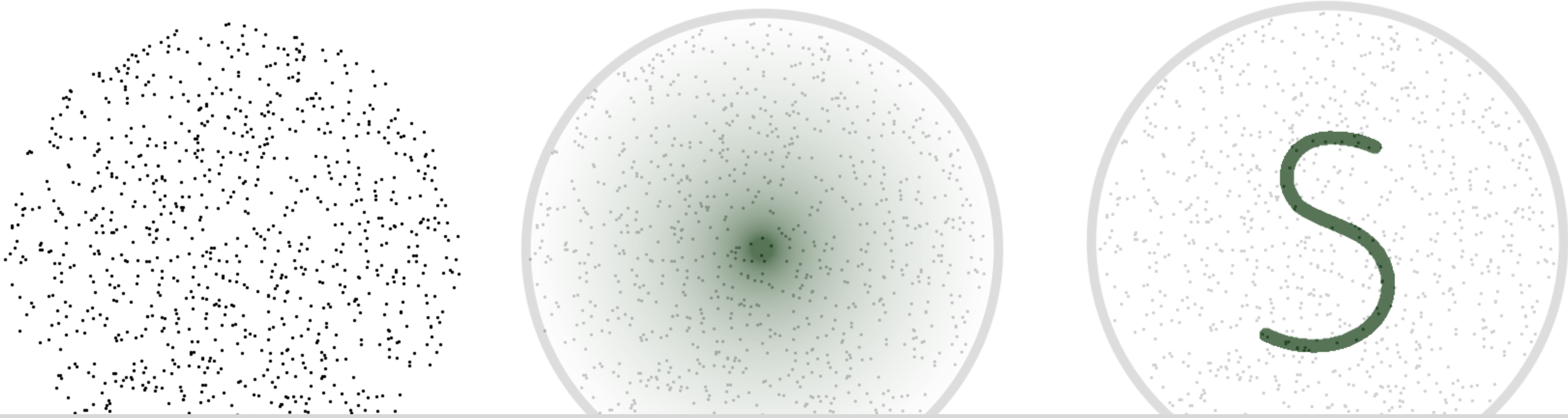
$$I = \int_{B(x)} G(x, z) f(z)\ dz$$

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

uniform samples

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

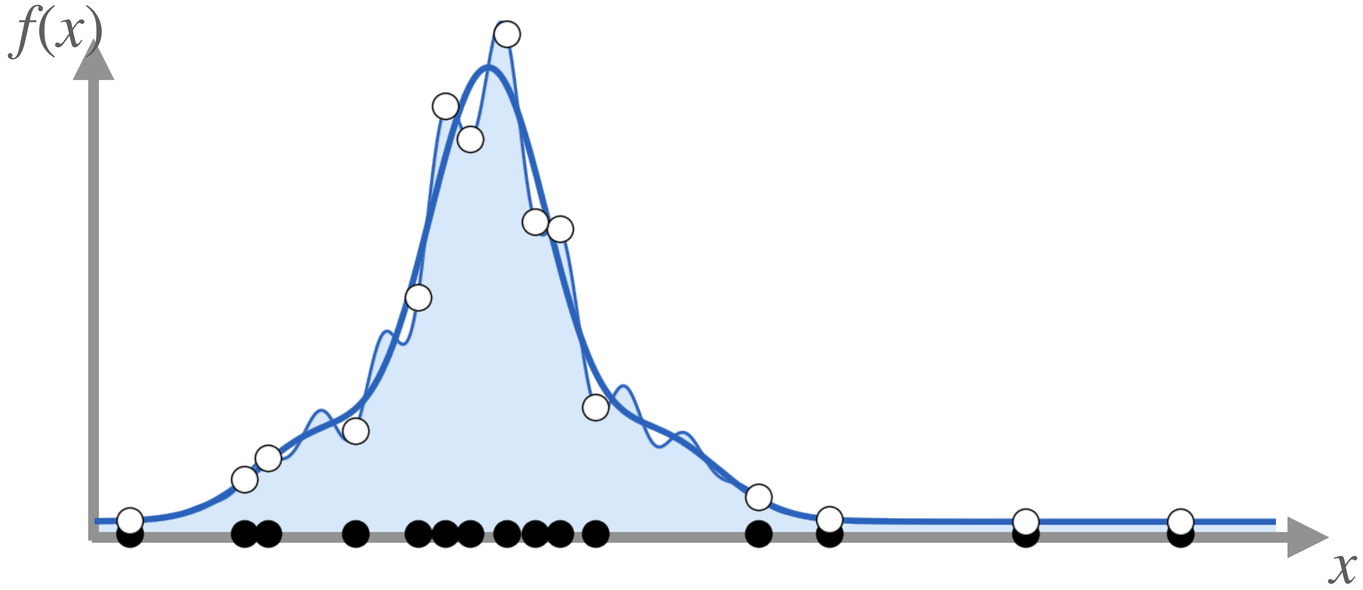
source function $f(z)$



Observation: most uniform samples don't contribute much…

Importance sampling to the rescue

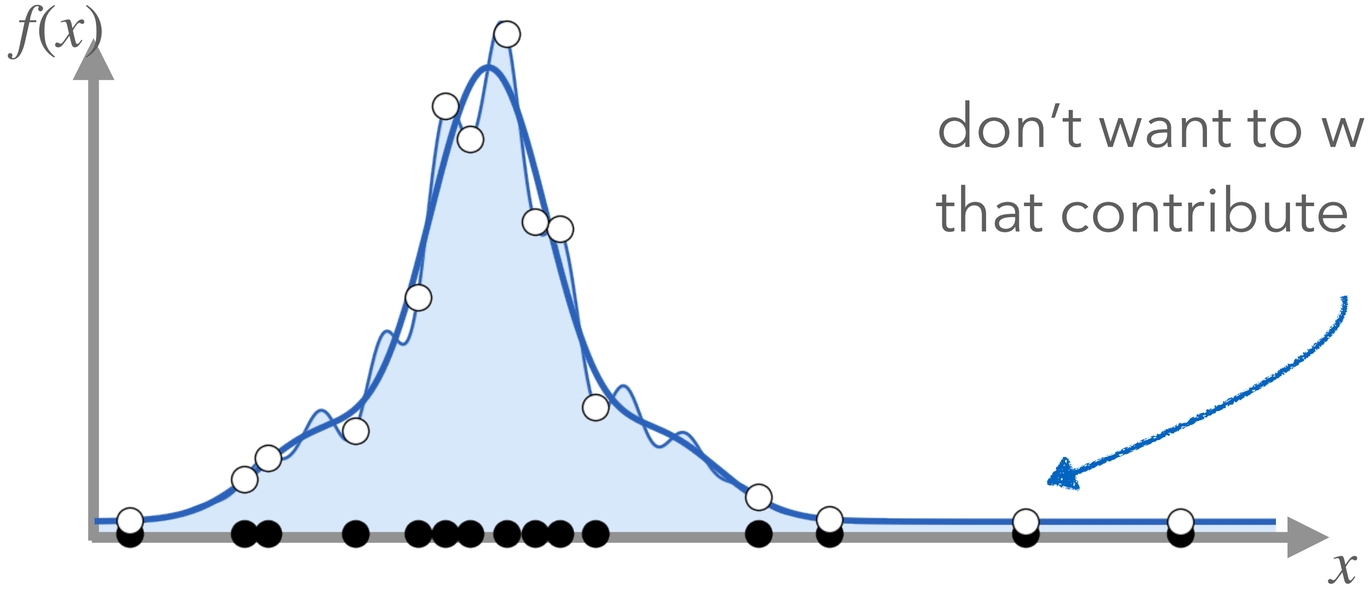
Idea: concentrate samples where the integrand is large



$$I = \int_a^b f(x)\ dx$$

Importance sampling to the rescue

Idea: concentrate samples where the integrand is large

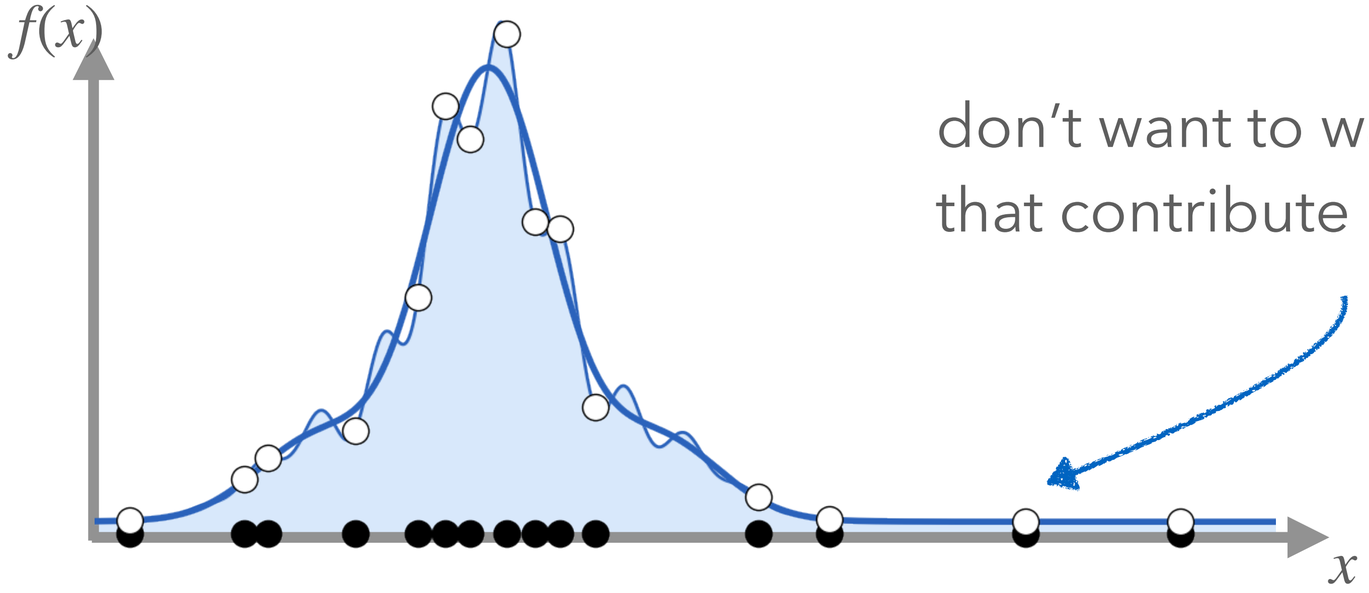


don’t want to waste time on points that contribute little to integral

$$I = \int_a^b f(x)\ dx$$

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

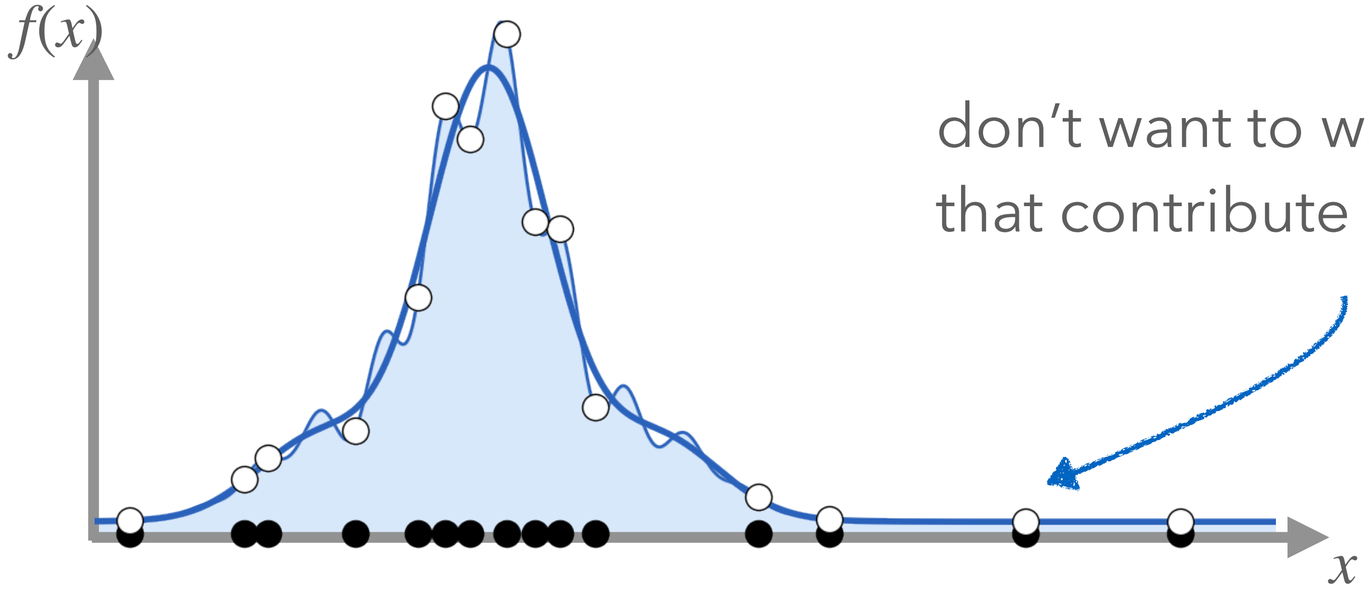
$$I = \int_a^b f(x)\ dx$$

draw more samples where p is large

$$X_i \sim p$$

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

$$I = \int_a^b f(x)\ dx$$

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

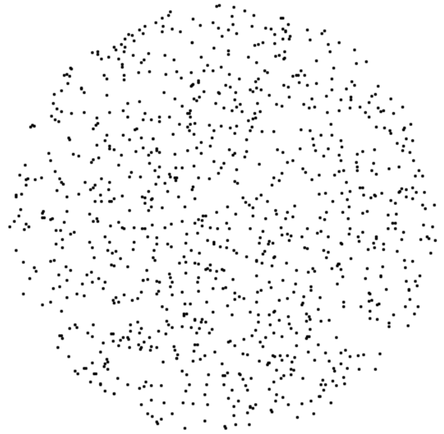
corrects for under / over sampling of uniform distribution

draw more samples where p is large

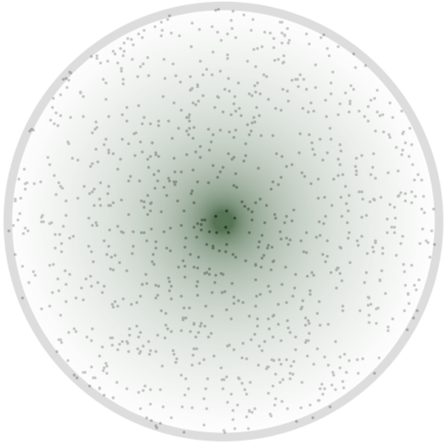
$$X_i \sim p$$

Sampling distributions: uniform vs Green's function

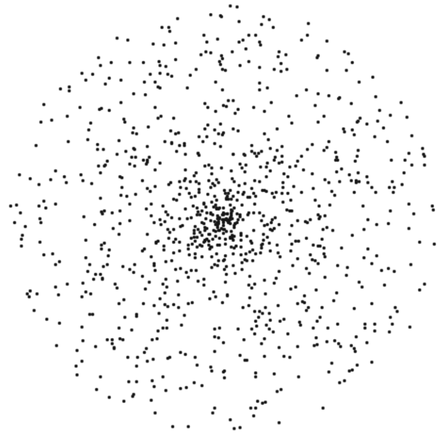
uniform samples



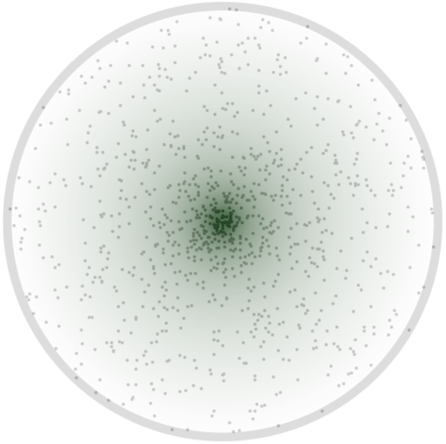
Green's function $G(x, z)$



importance samples



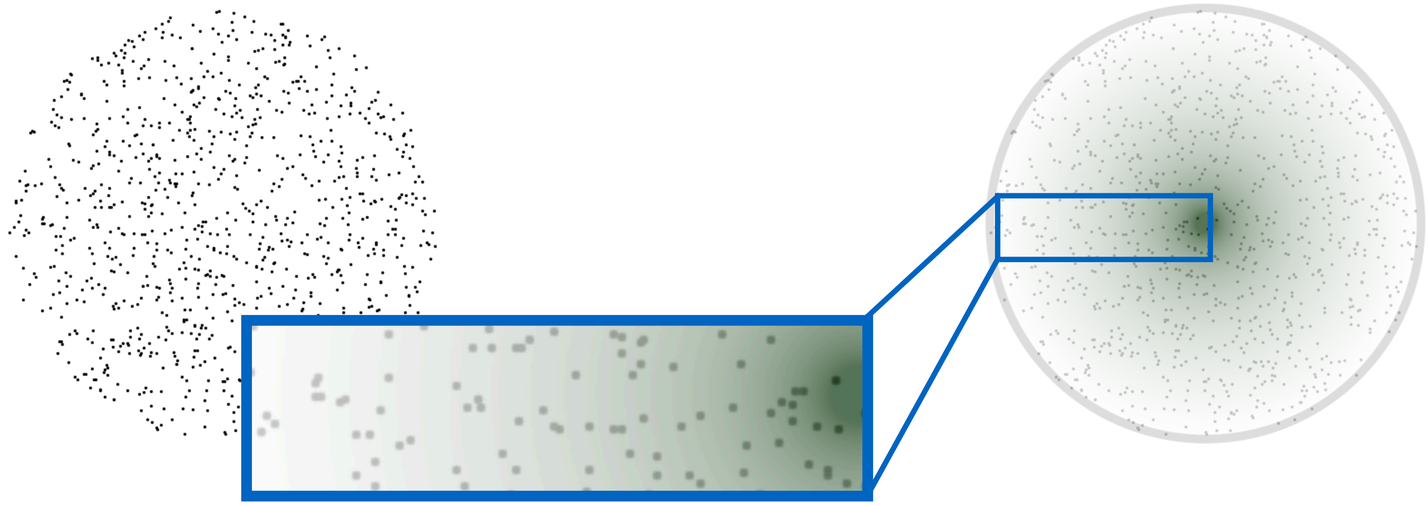
Green's function $G(x, z)$



Sampling distributions: uniform vs Green's function

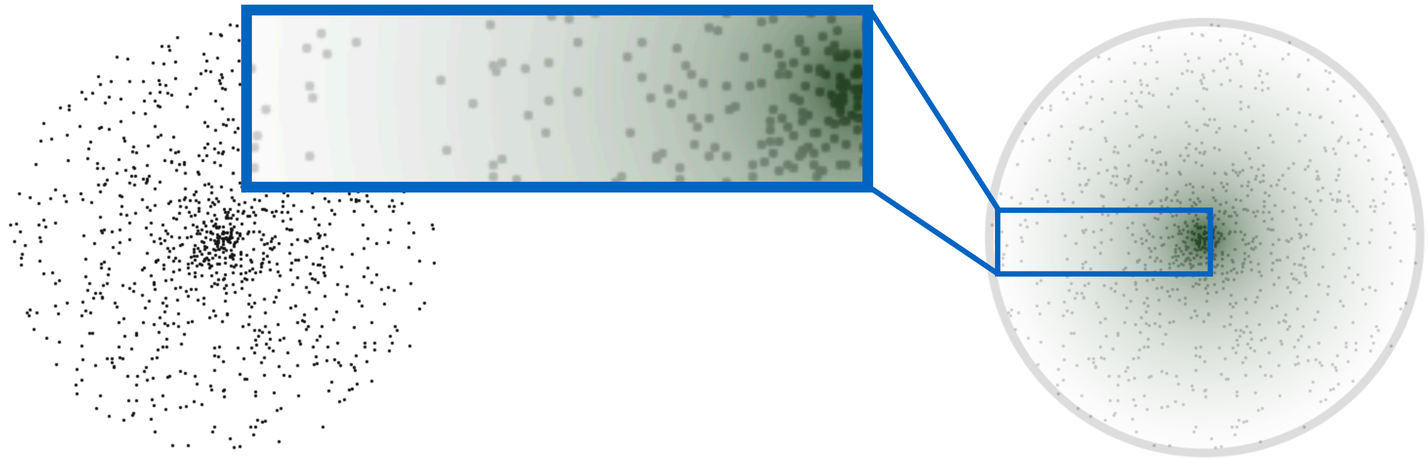
uniform samples

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

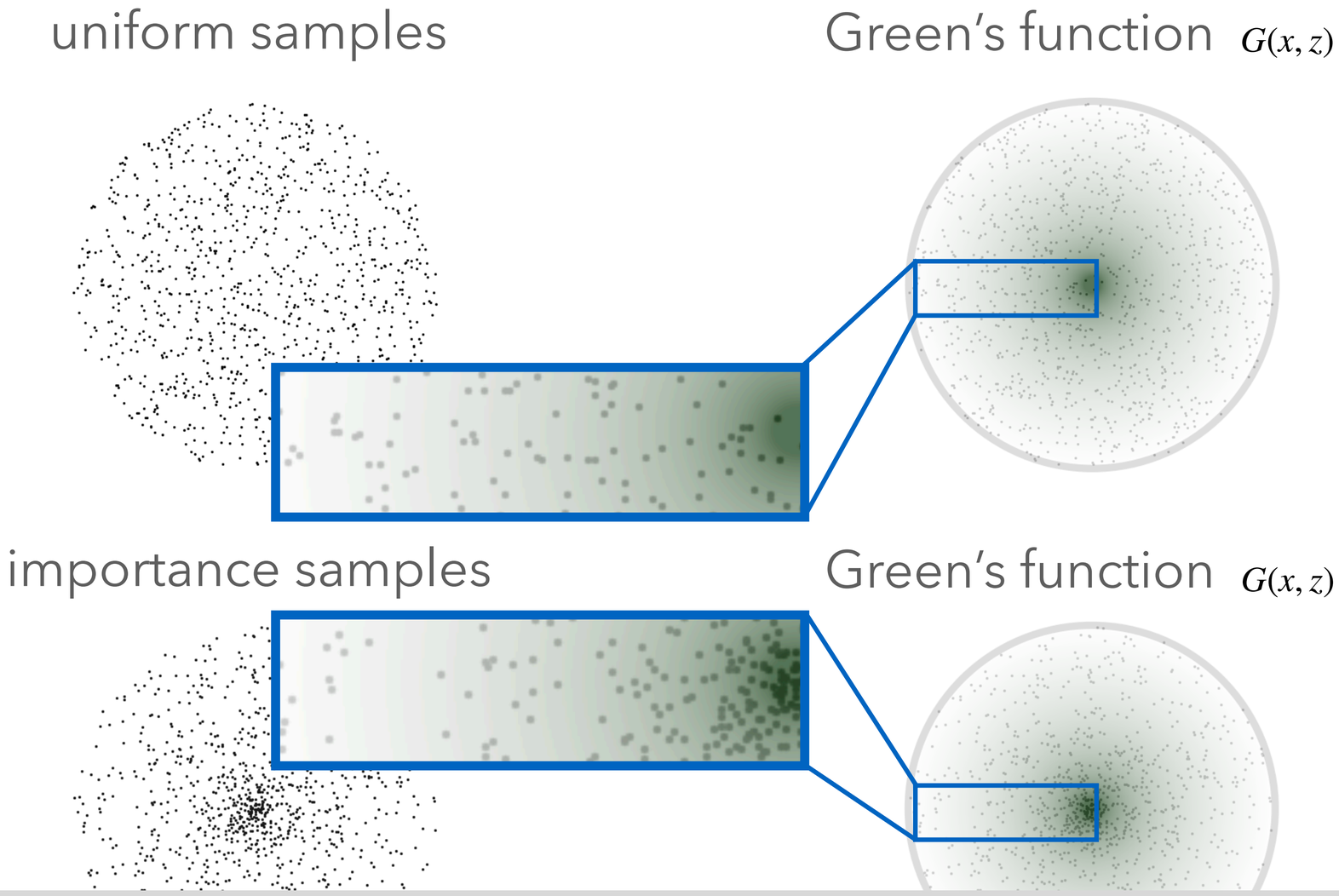


importance samples

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



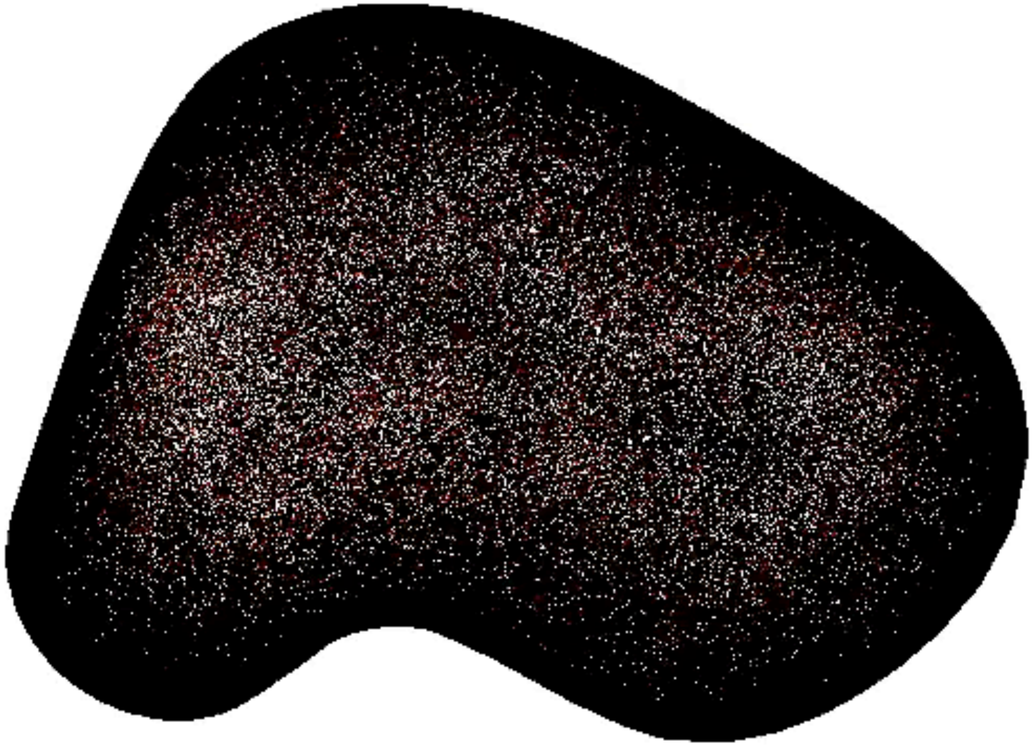
Sampling distributions: uniform vs Green's function



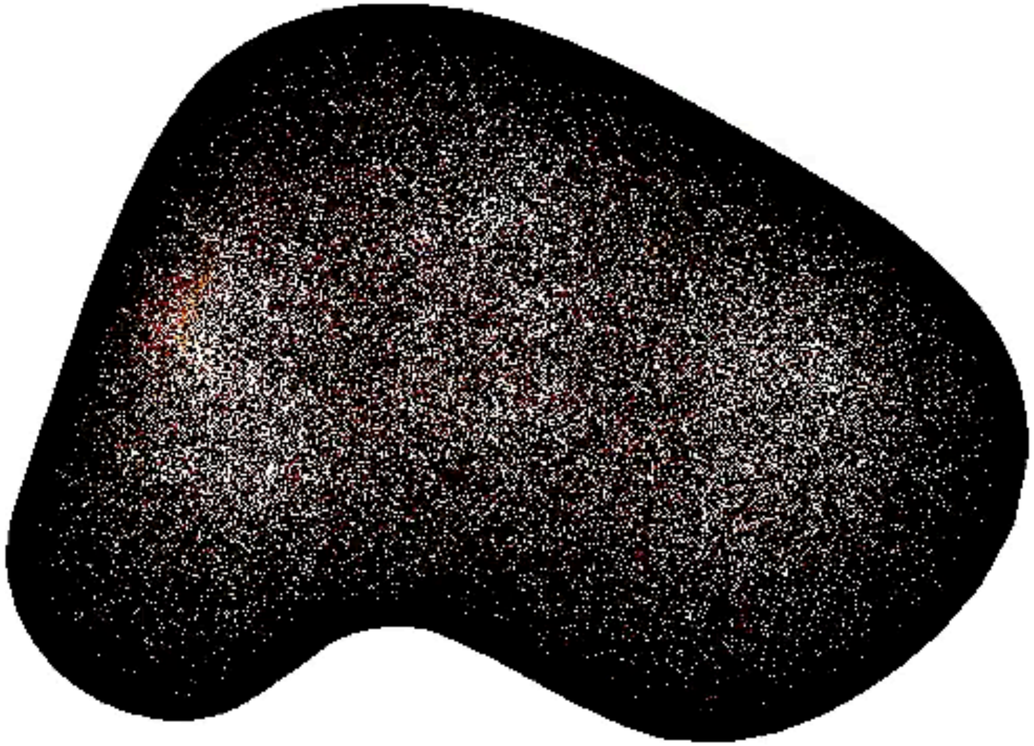
key: importance sampling improves efficiency

Walk on spheres with and without importance sampling

uniform sampling

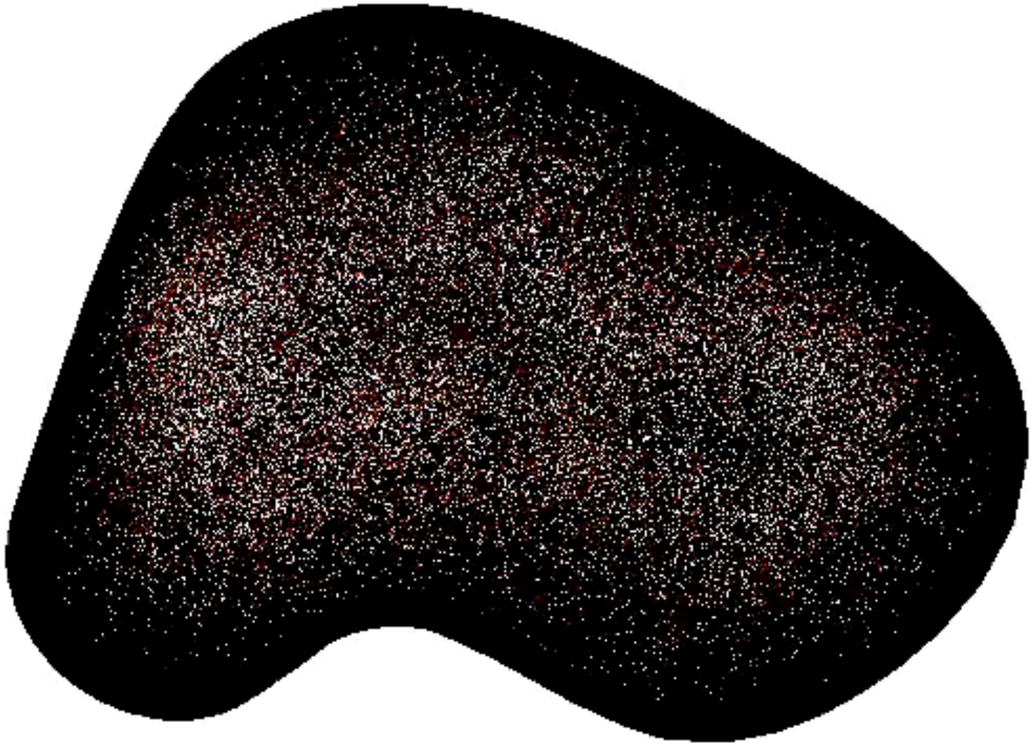


importance sampling Green’s function

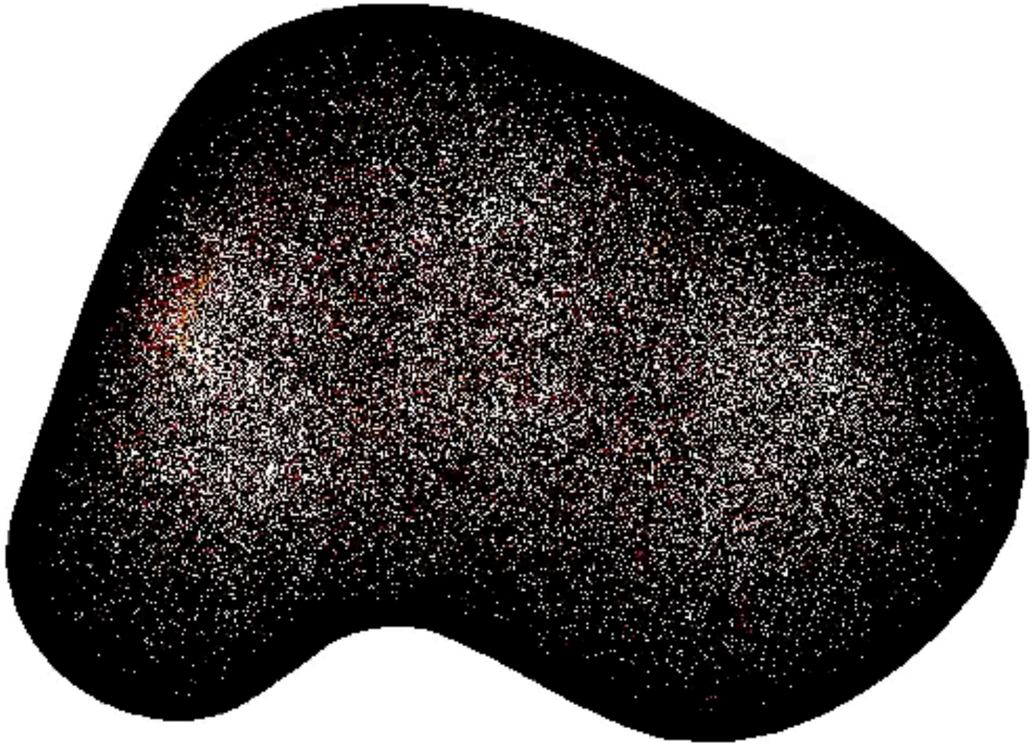


Walk on spheres with and without importance sampling

uniform sampling

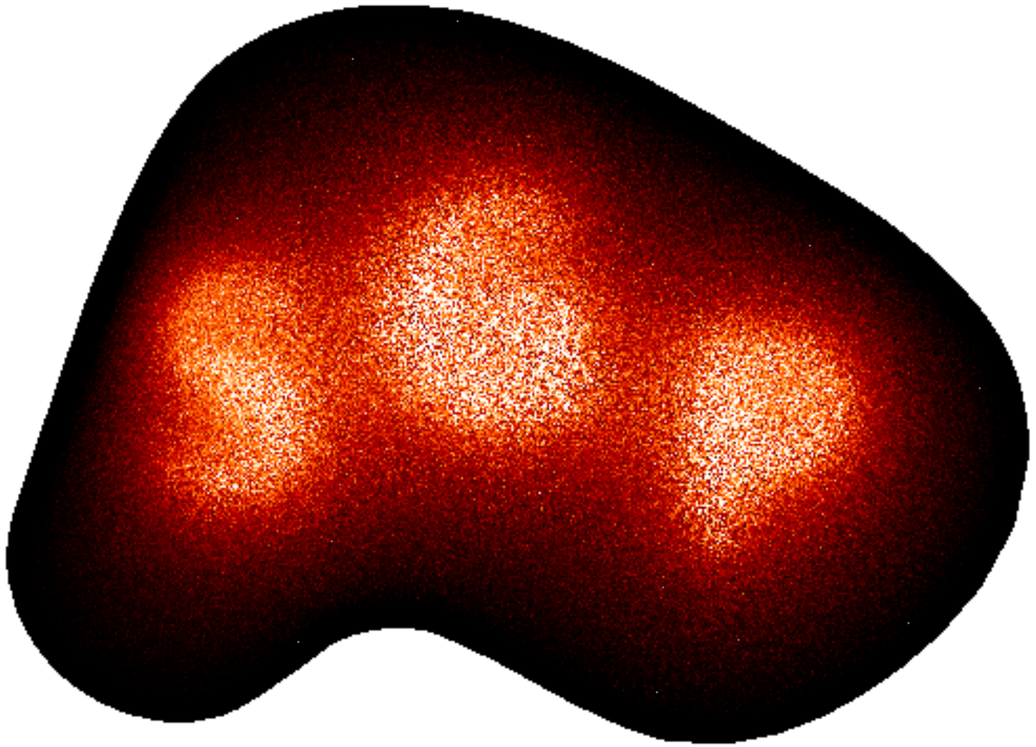


importance sampling Green’s function

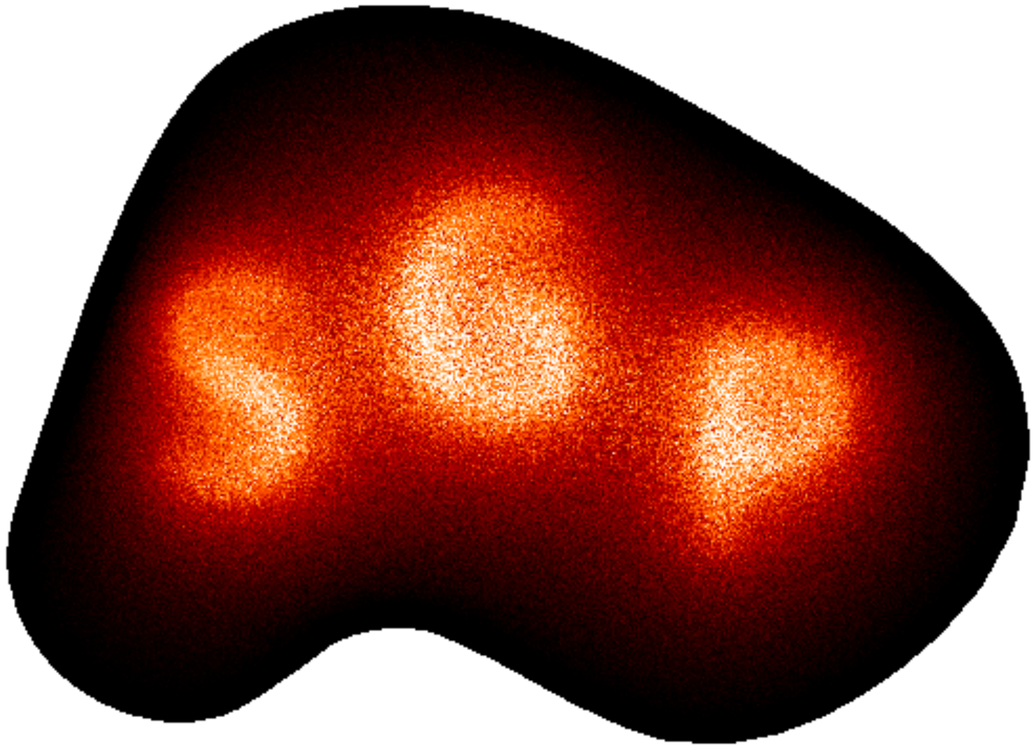


Walk on spheres with and without importance sampling

uniform sampling

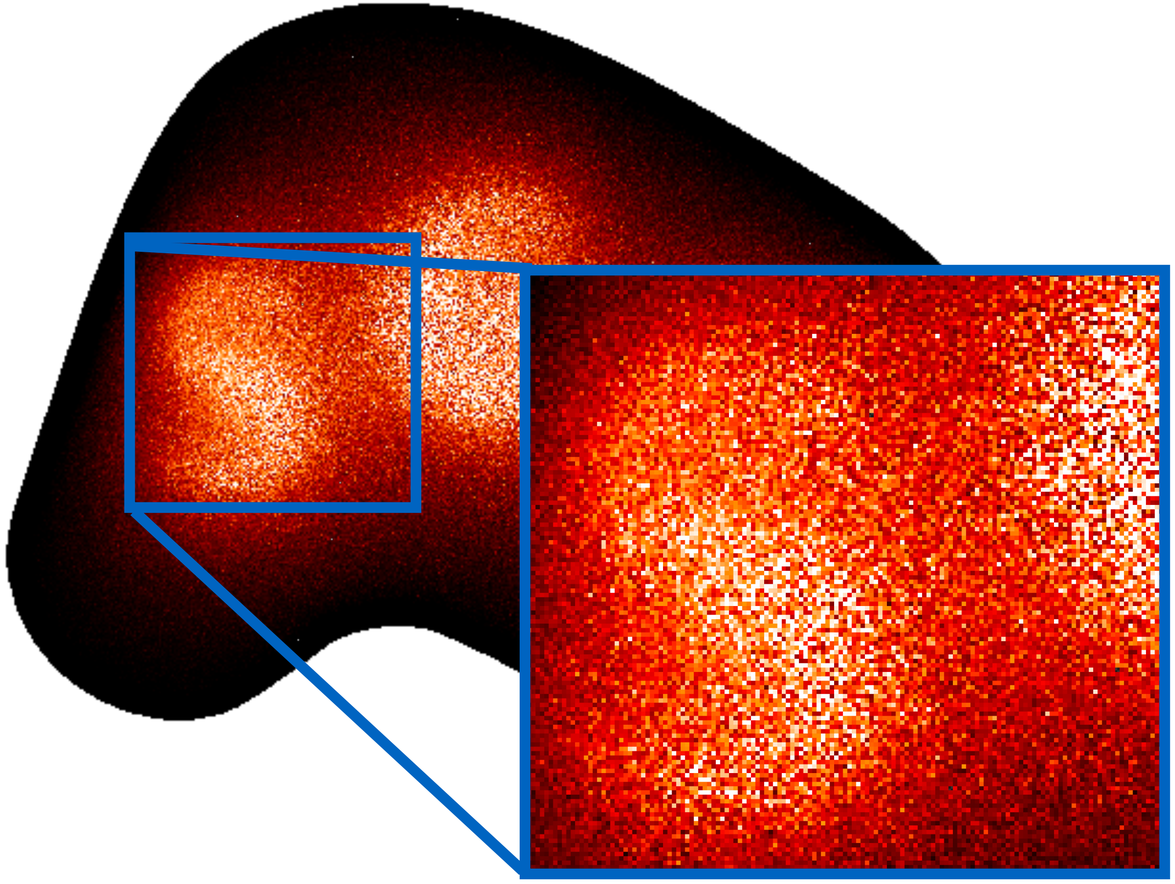


importance sampling Green's function

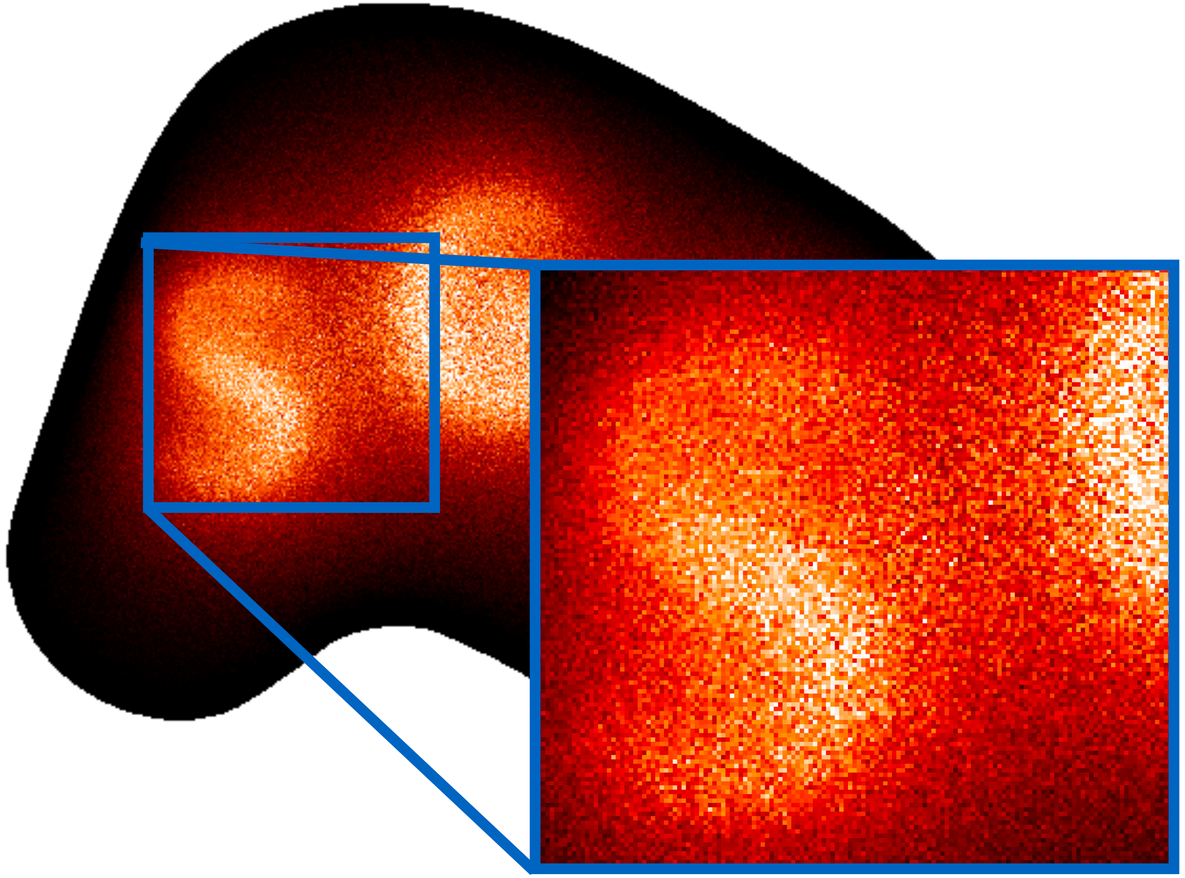


Walk on spheres with and without importance sampling

uniform sampling

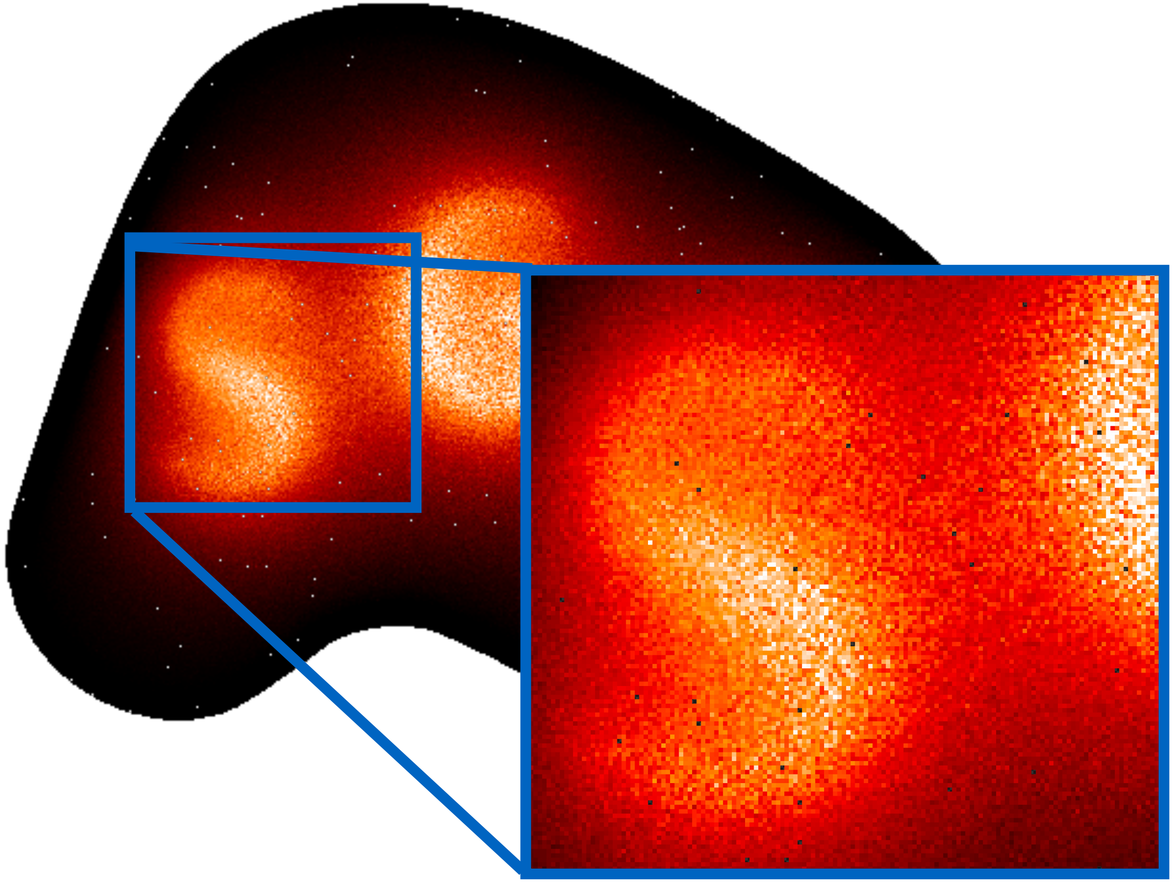


importance sampling Green's function

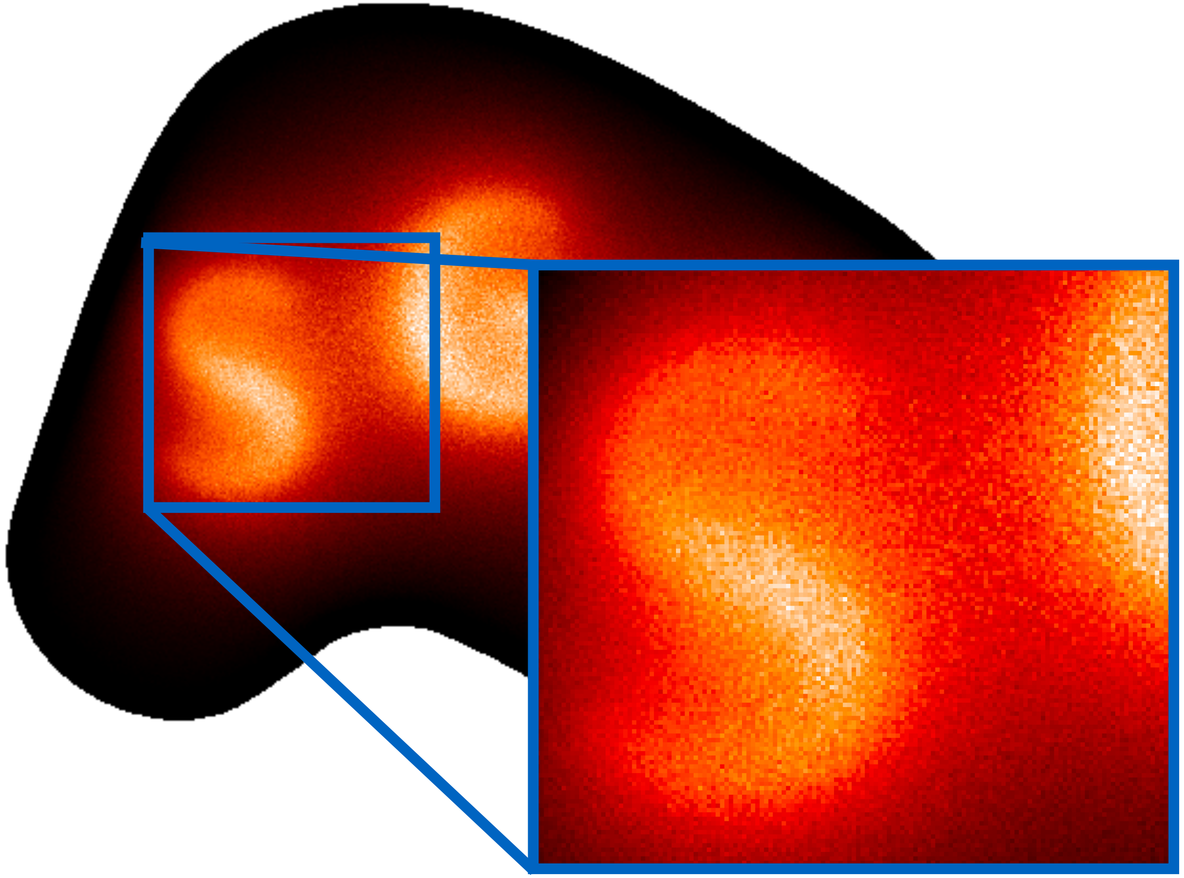


Walk on spheres with and without importance sampling

uniform sampling



importance sampling Green's function



Importance sampling additional terms

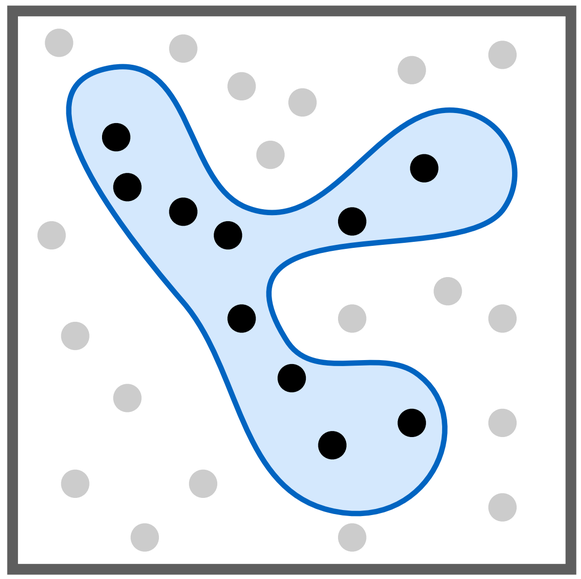
uniform samples

source samples $f(z)$

How do we actually sample points?

How do we actually sample points?

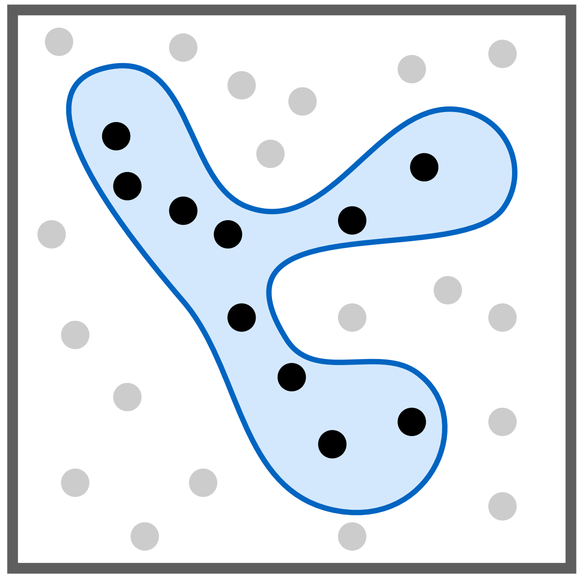
rejection sampling



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

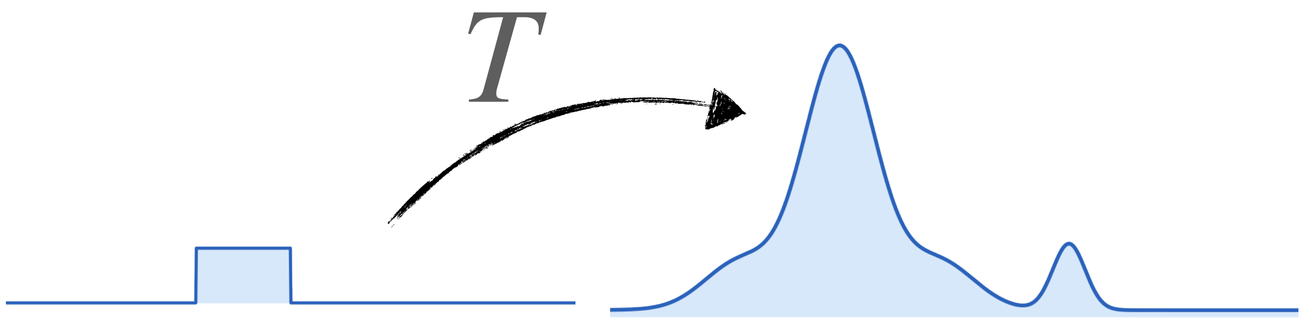
How do we actually sample points?

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

Combining sampling strategies

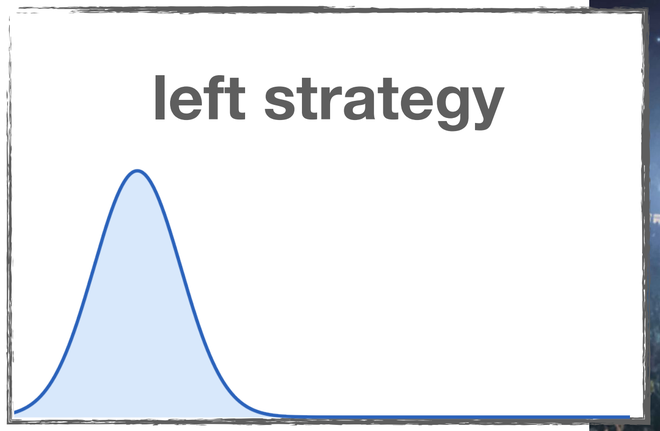
- In practice, may have more than one importance sampling strategy that seems promising. Which one should you use?

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

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



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

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

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

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 3:
Variance Reduction

Efficiency of Monte Carlo

Squared error of Monte Carlo estimator is given by:

Efficiency of Monte Carlo

Squared error of Monte Carlo estimator is given by:

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

Efficiency of Monte Carlo

Squared error of Monte Carlo estimator is given by:

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

$V[f]$ ← variance of the integrand

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

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:

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

Efficiency of Monte Carlo

Squared error of Monte Carlo estimator is given by:

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

$V[f]$ ← variance of the integrand

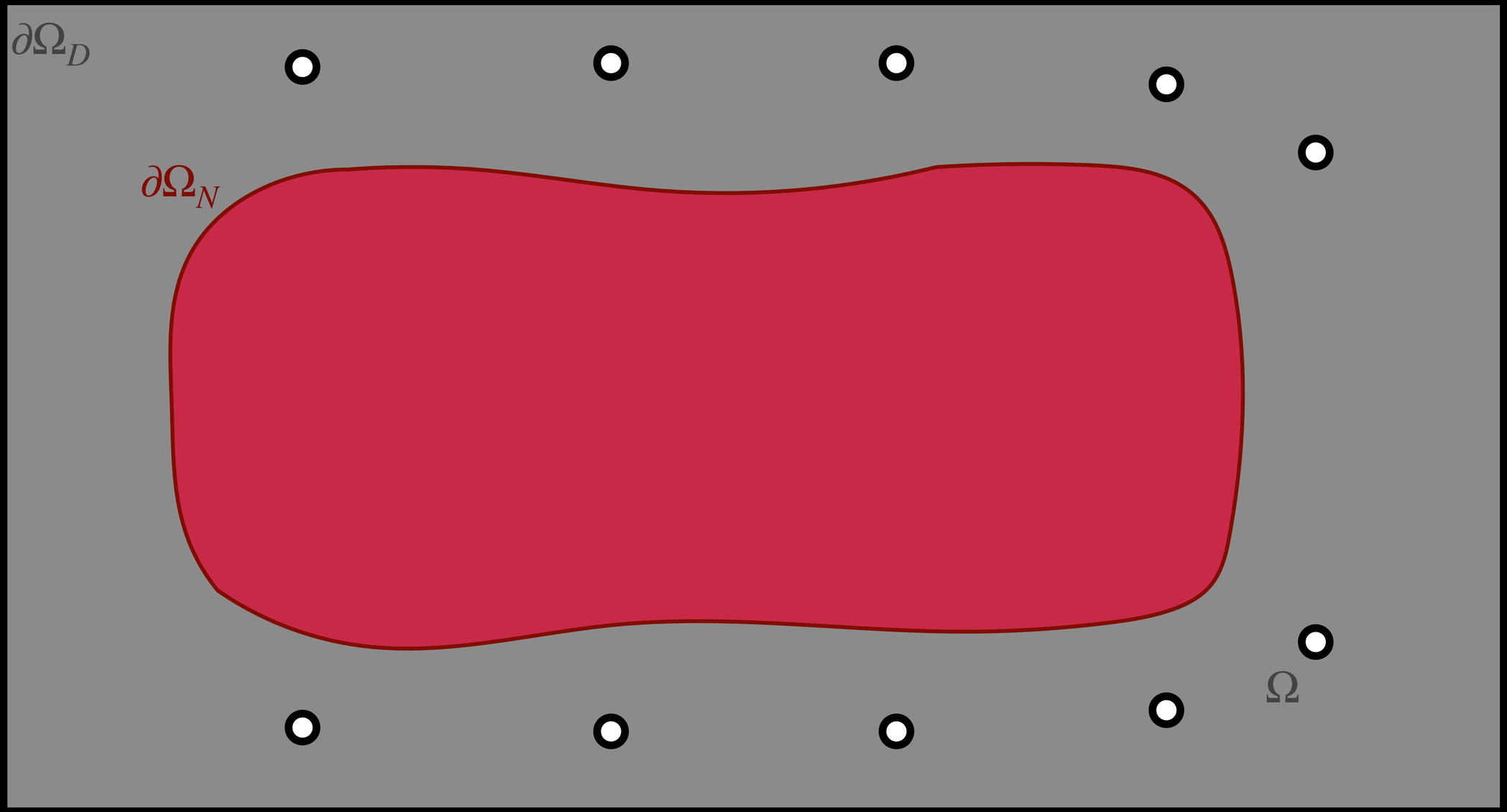
N ← number of samples

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

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

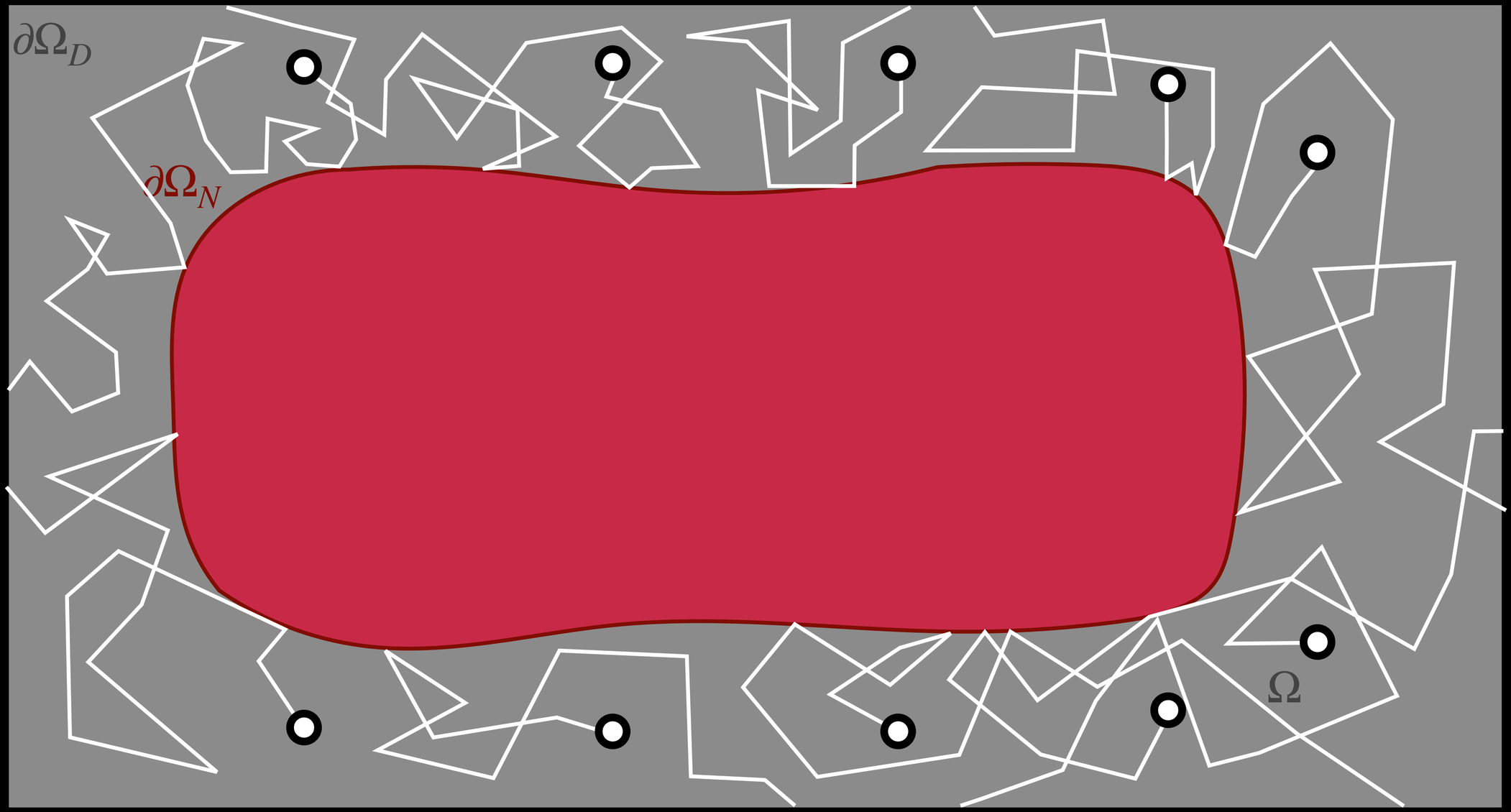
– **more samples per second ($1/N$)**, e.g., parallelism, GPUs, caching

Solution evaluated independently at very point – parallelism



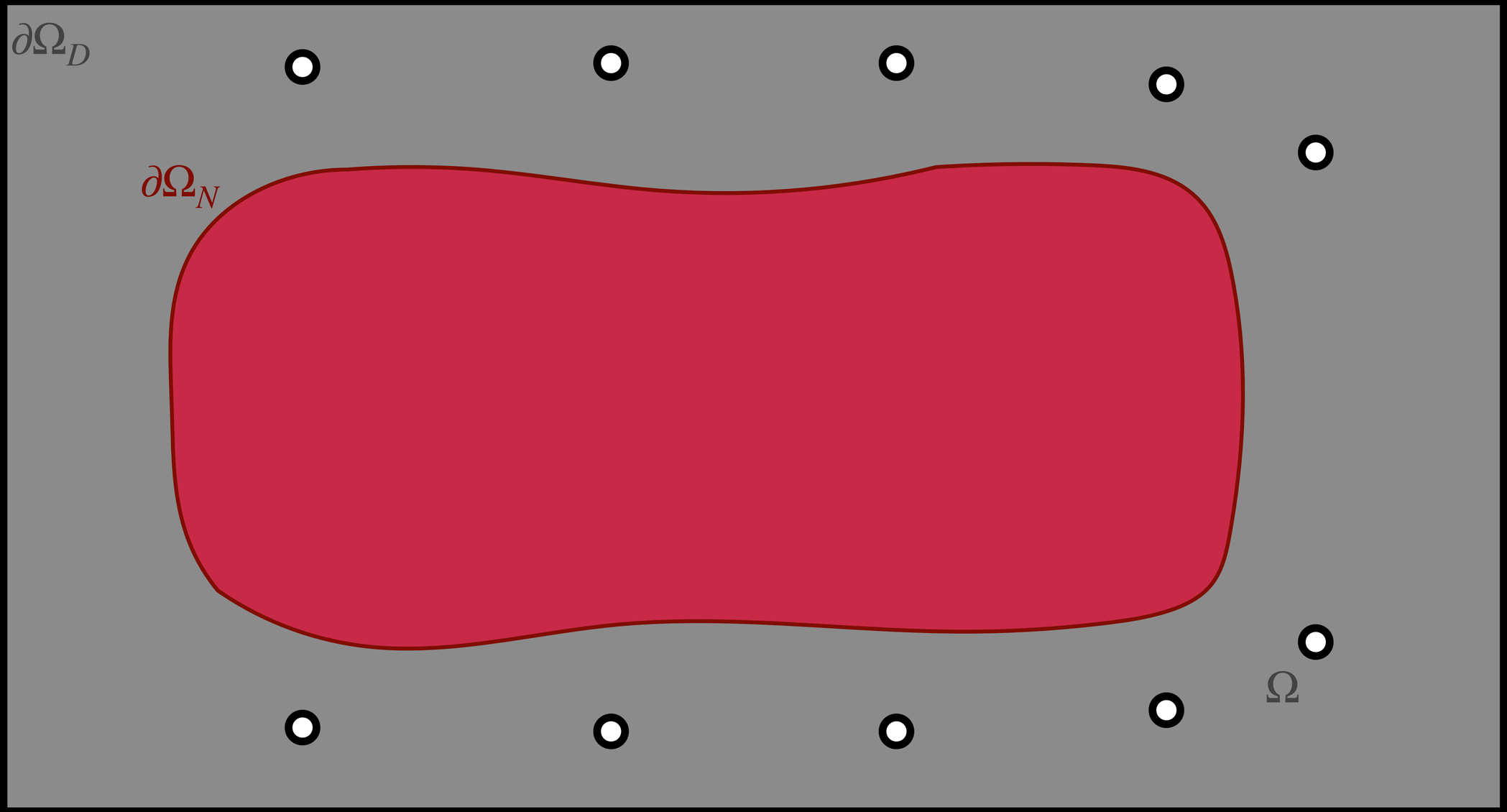
walk on stars

Solution evaluated independently at very point – parallelism



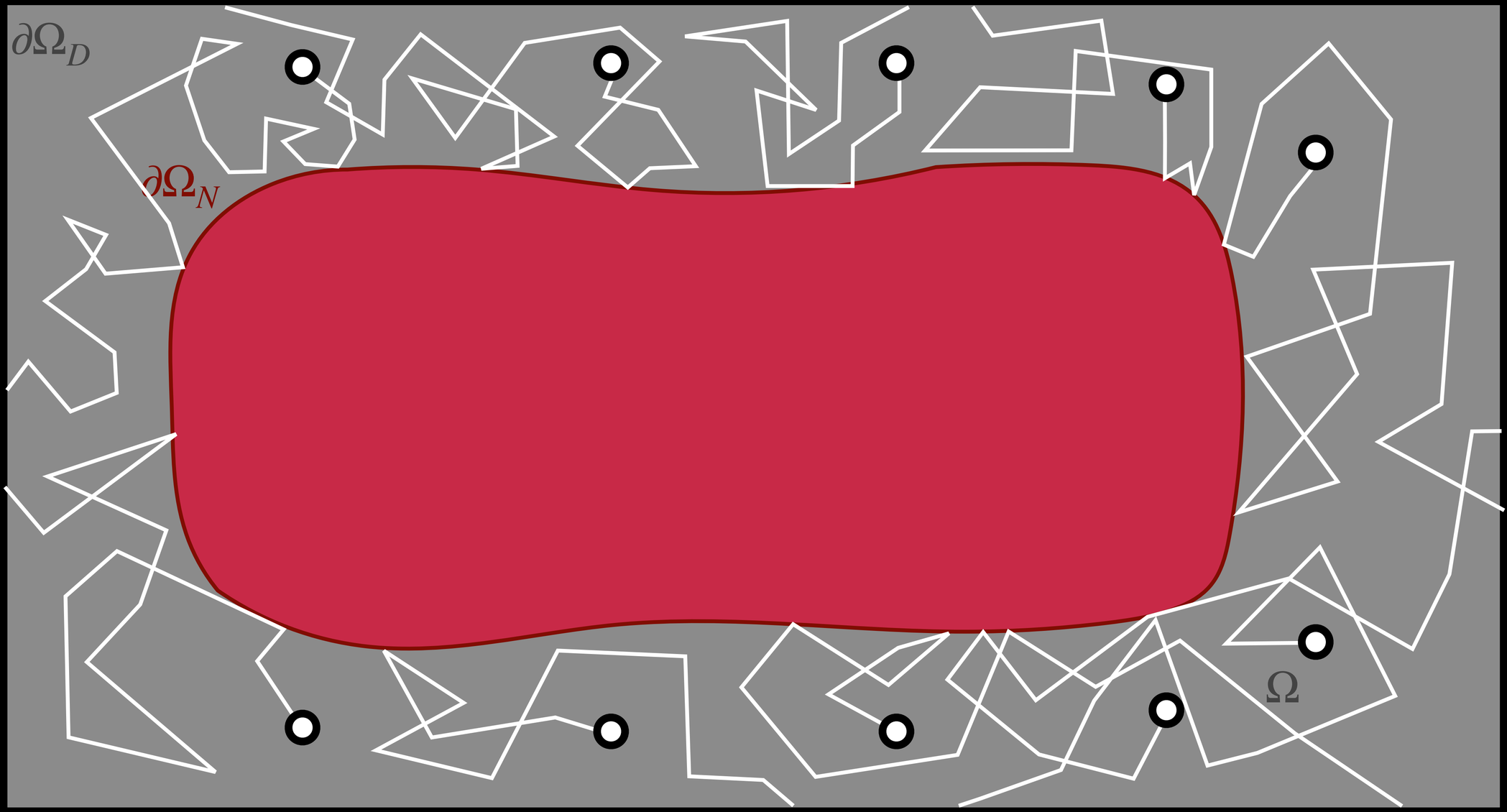
walk on stars

Solution evaluated independently at very point – redundancy



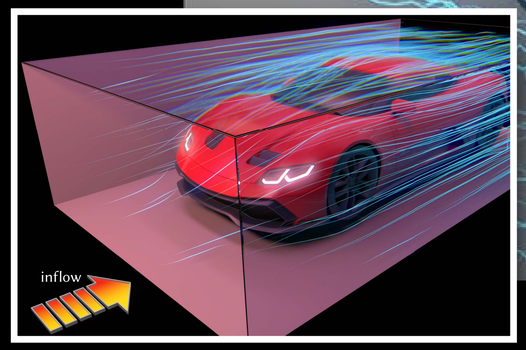
walk on stars

Solution evaluated independently at very point – redundancy



walk on stars

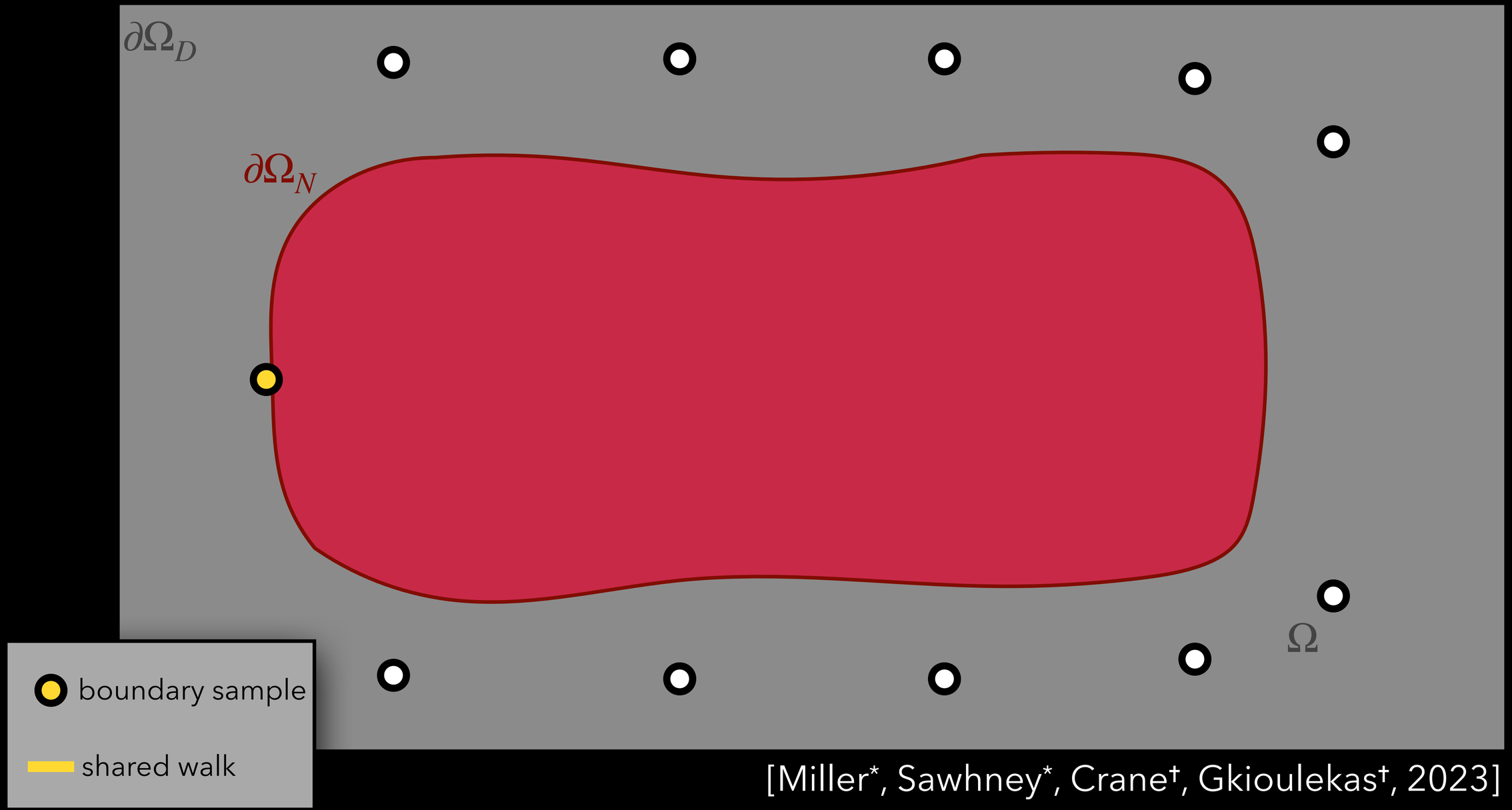
Noisy streamlines!



wind tunnel

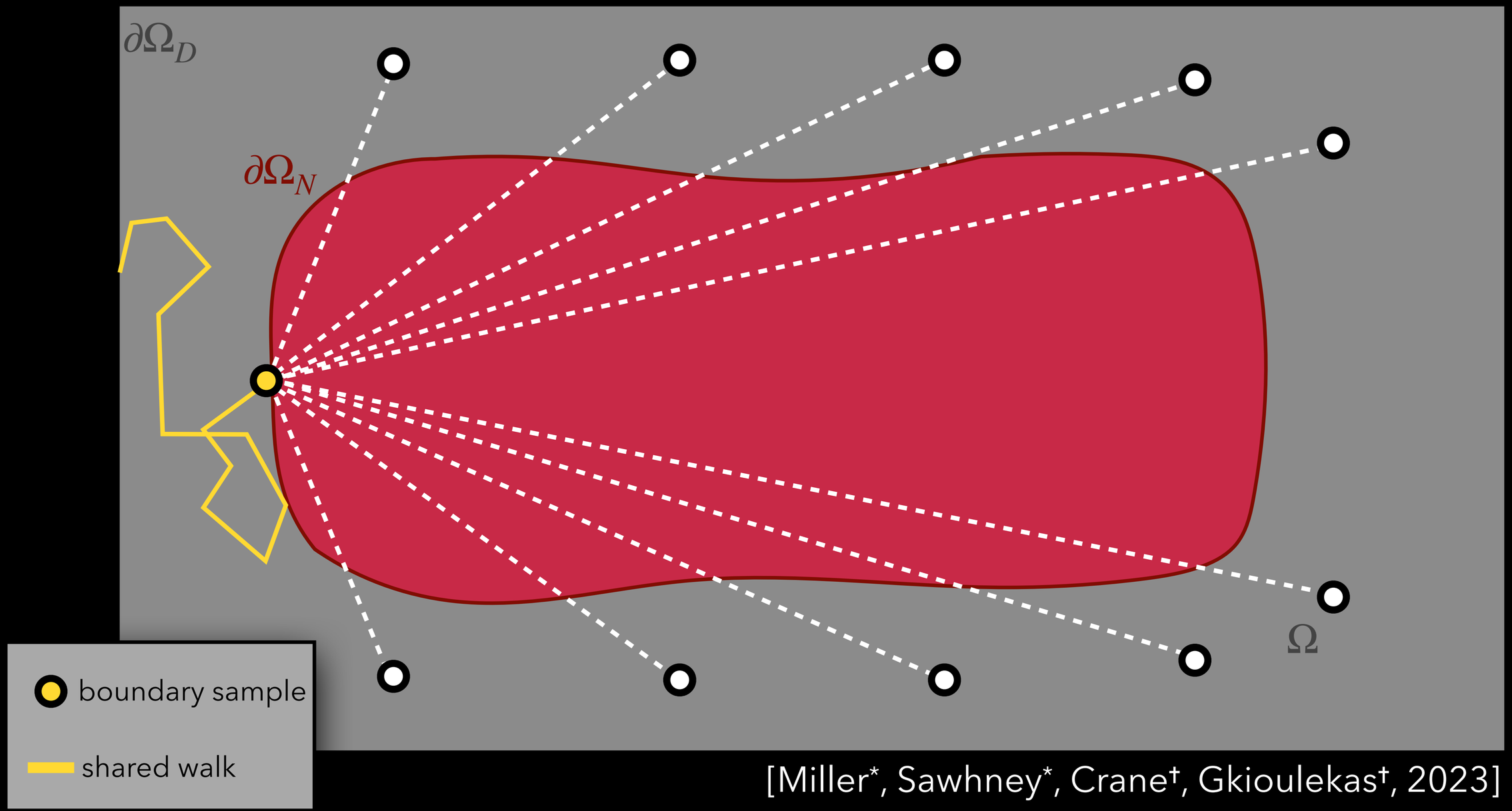
walk on stars

boundary value caching

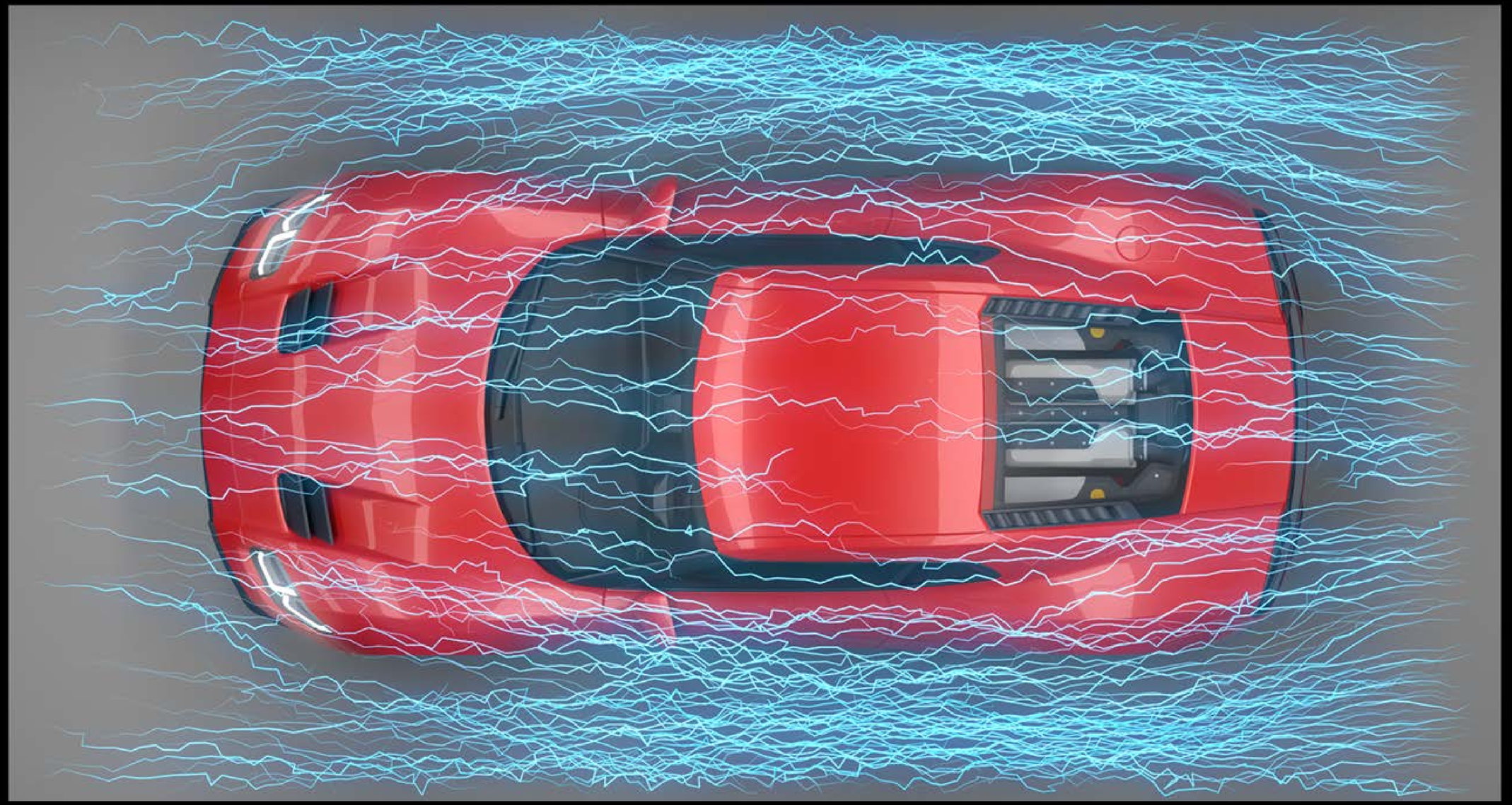


[Miller*, Sawhney*, Crane†, Gkioulekas†, 2023]

boundary value caching

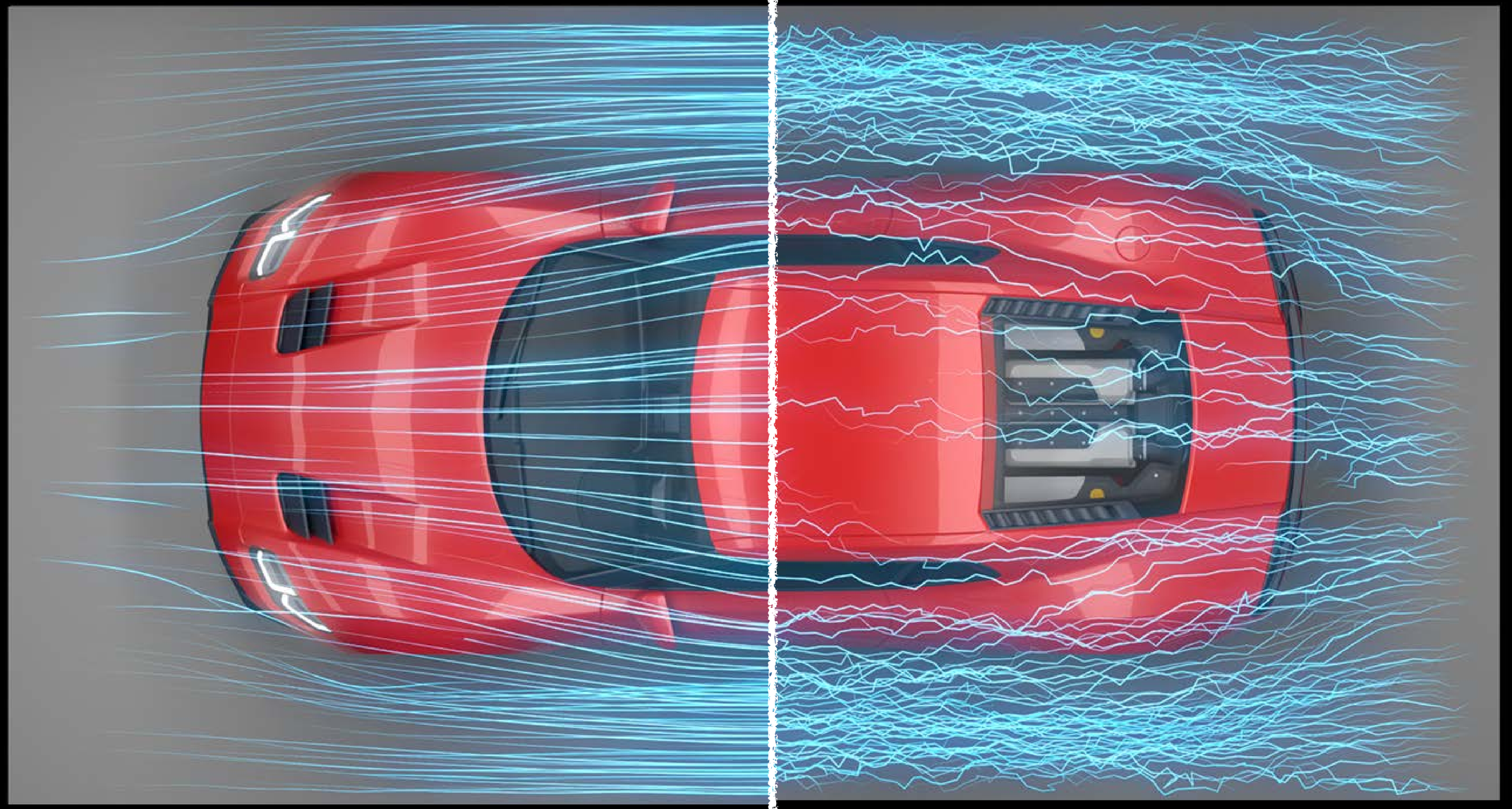


[Miller*, Sawhney*, Crane†, Gkioulekas†, 2023]

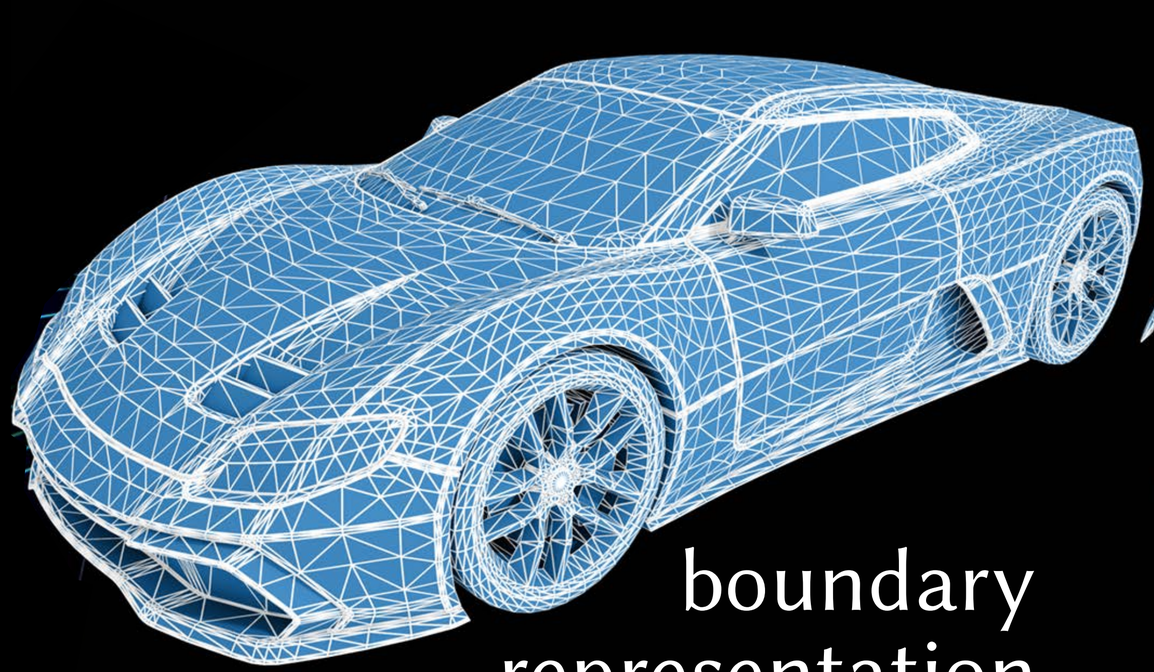


boundary value caching

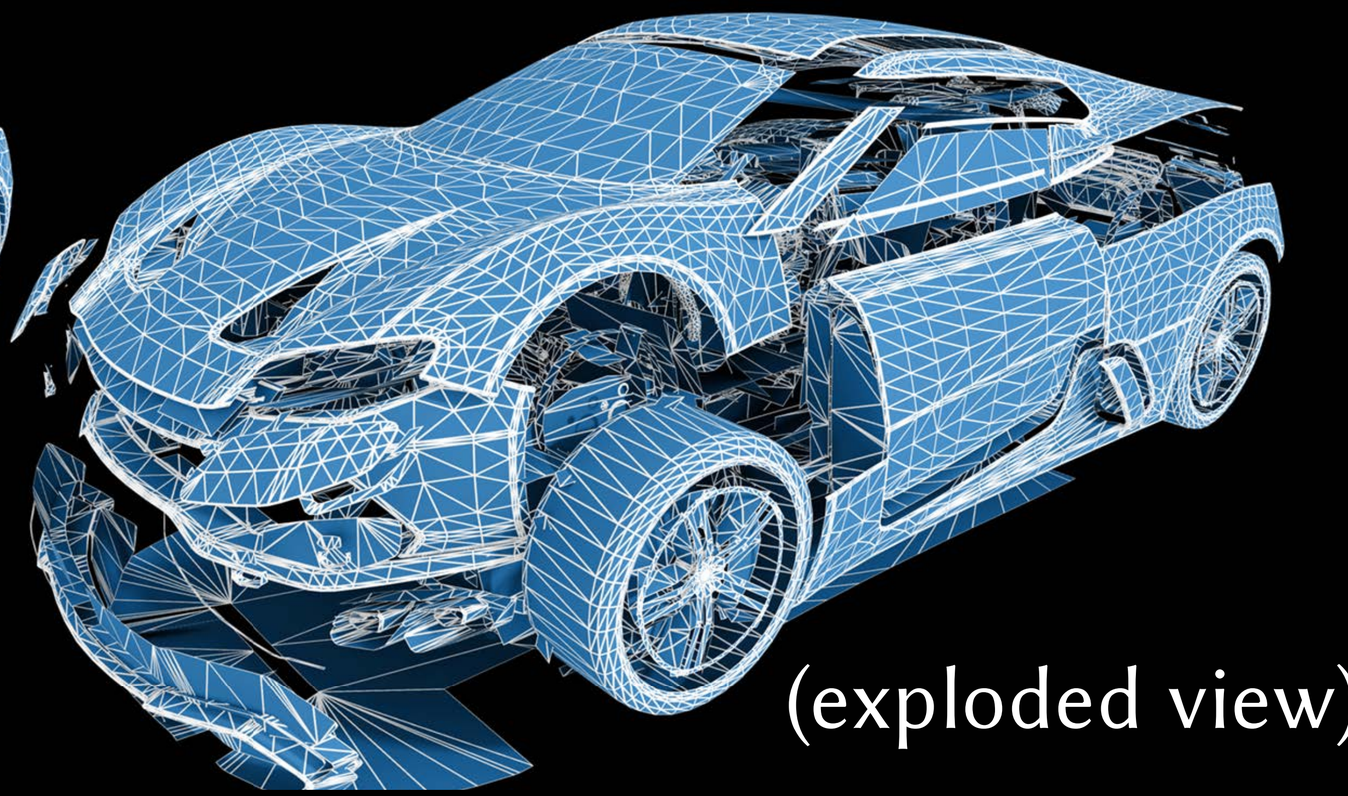
walk on stars


boundary value caching
walk on stars

Robustly handle meshes intended for visualization



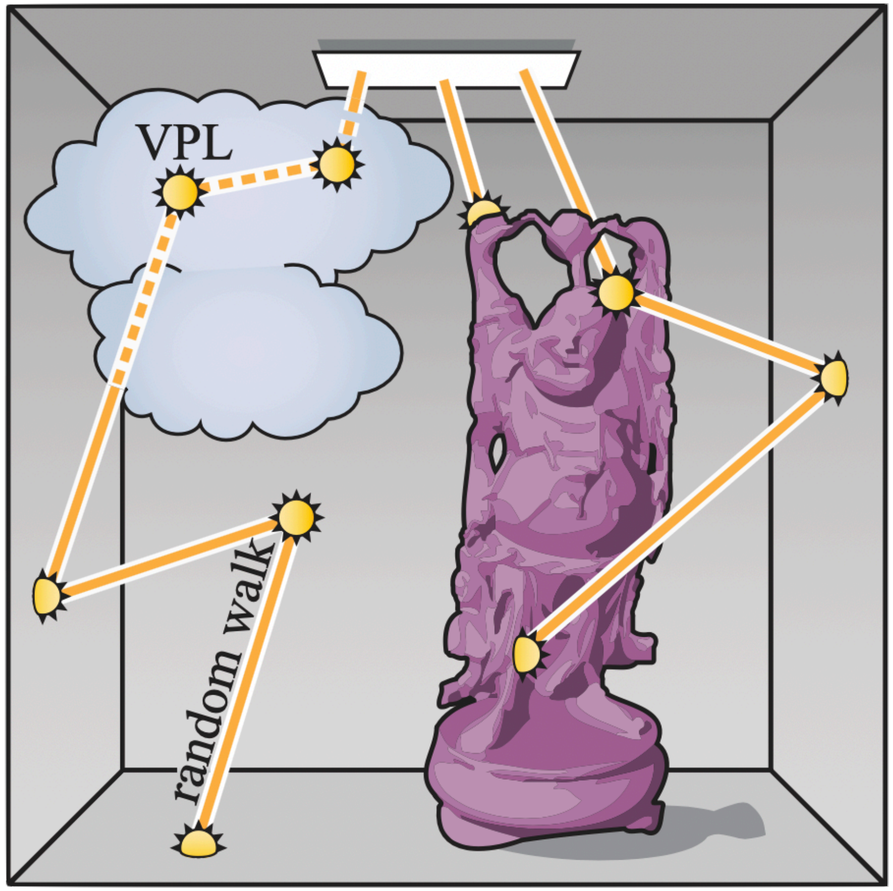
boundary
representation



(exploded view)

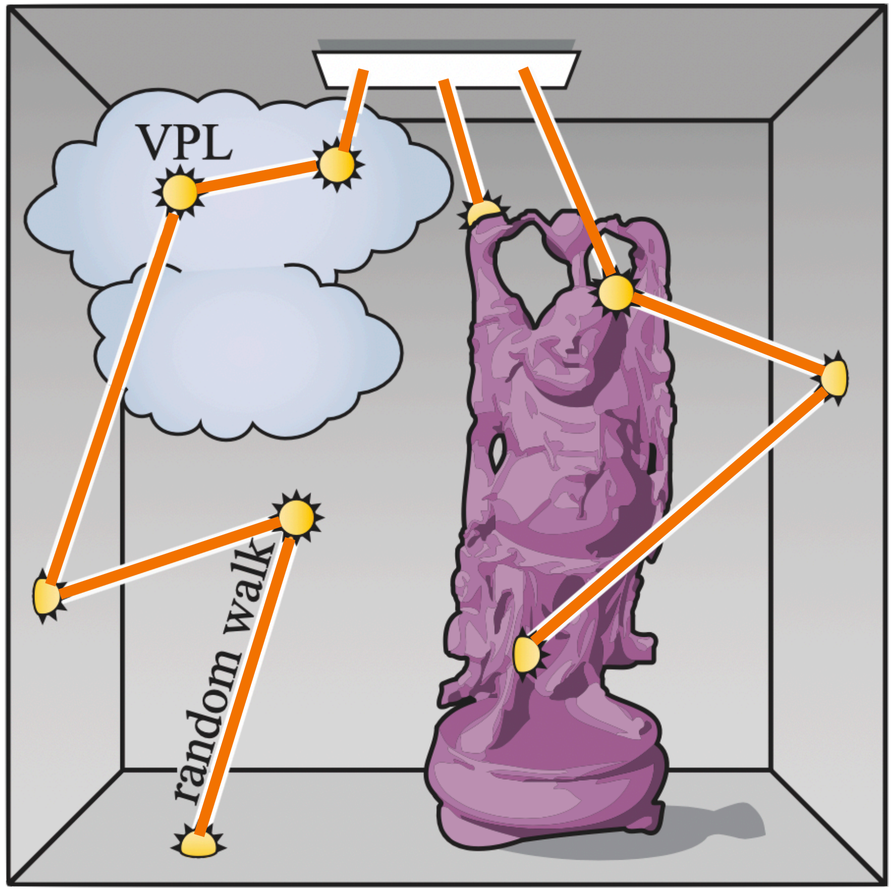
Sample reuse in Monte Carlo Ray Tracing

Virtual Point Light Methods (VPLs)



Sample reuse in Monte Carlo Ray Tracing

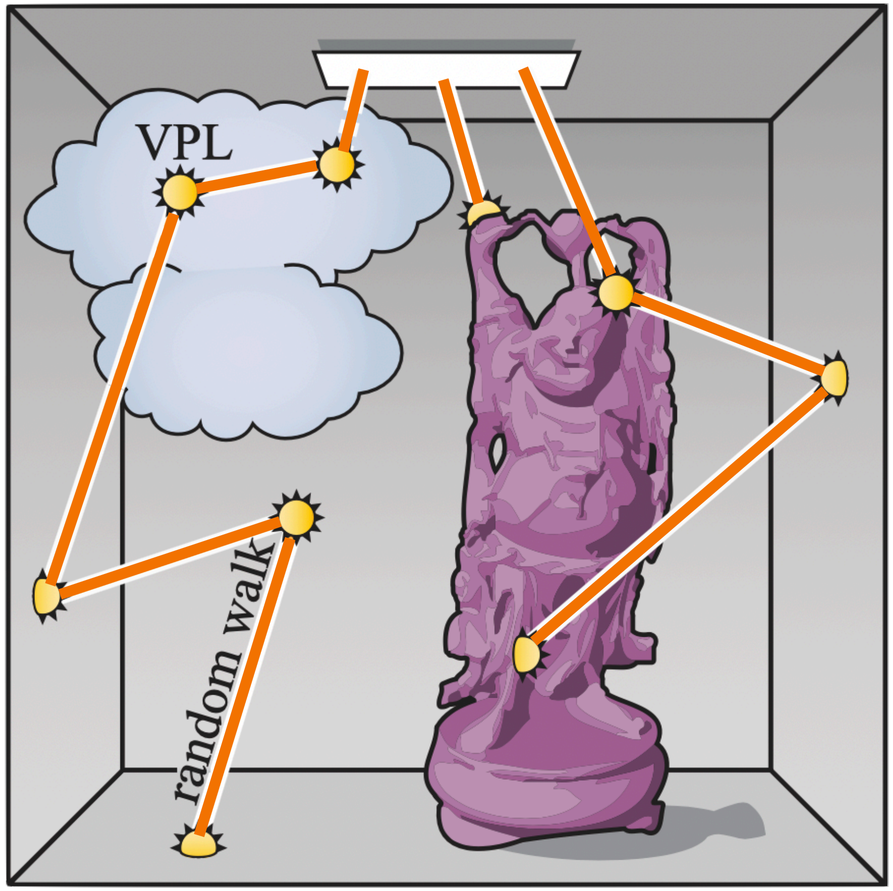
Virtual Point Light Methods (VPLs)



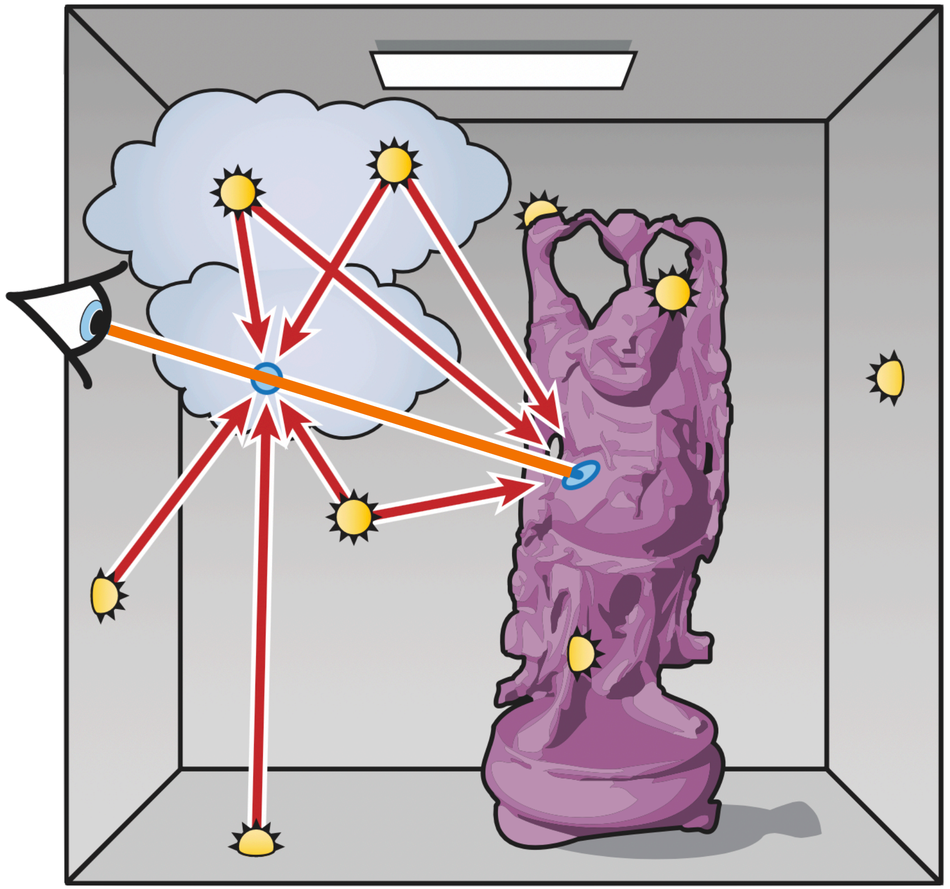
Step 1: Deposit radiance estimates

Sample reuse in Monte Carlo Ray Tracing

Virtual Point Light Methods (VPLs)



Step 1: Deposit radiance estimates



Step 2: Reuse cached radiance estimates

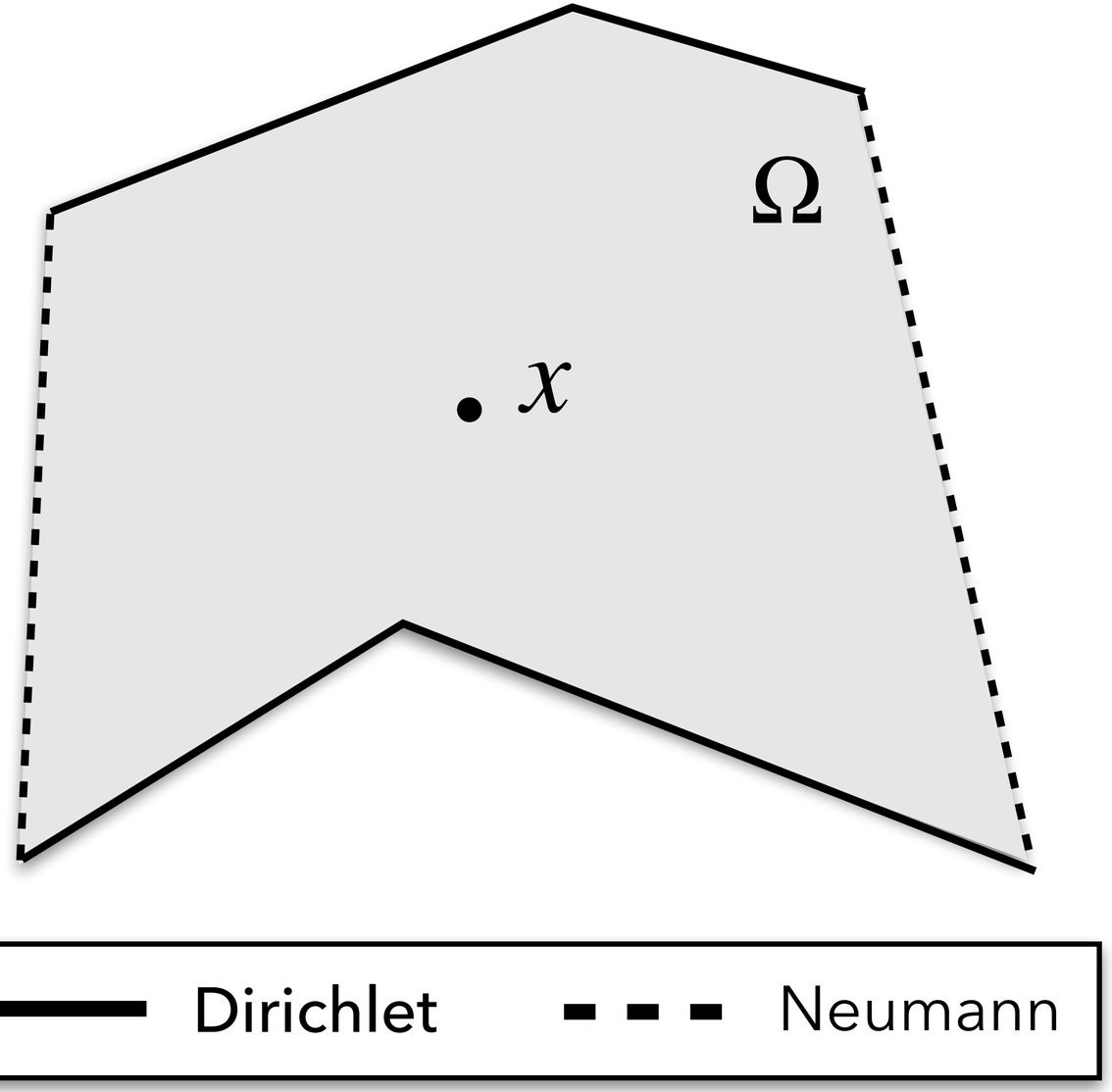
Sample reuse for PDEs

Laplace equation

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

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

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



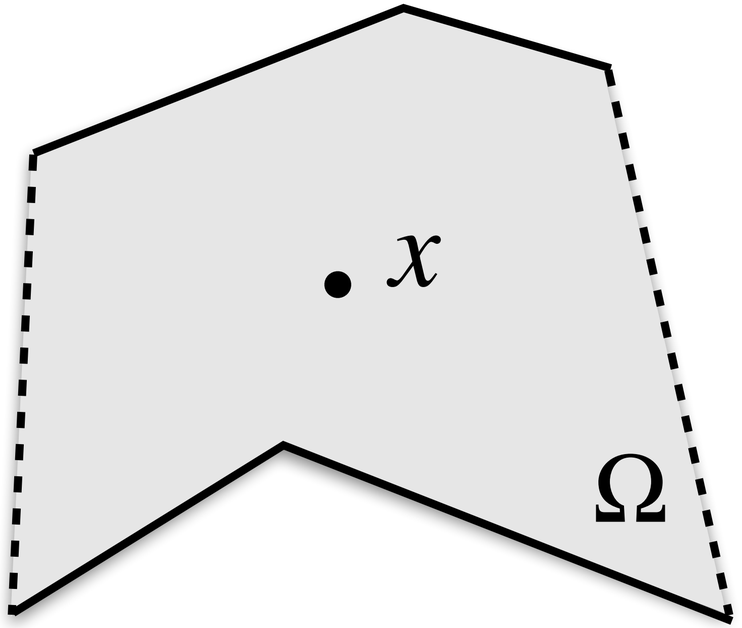
Boundary Integral Equation

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

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

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

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



Boundary Integral Equation

free-space Poisson kernel

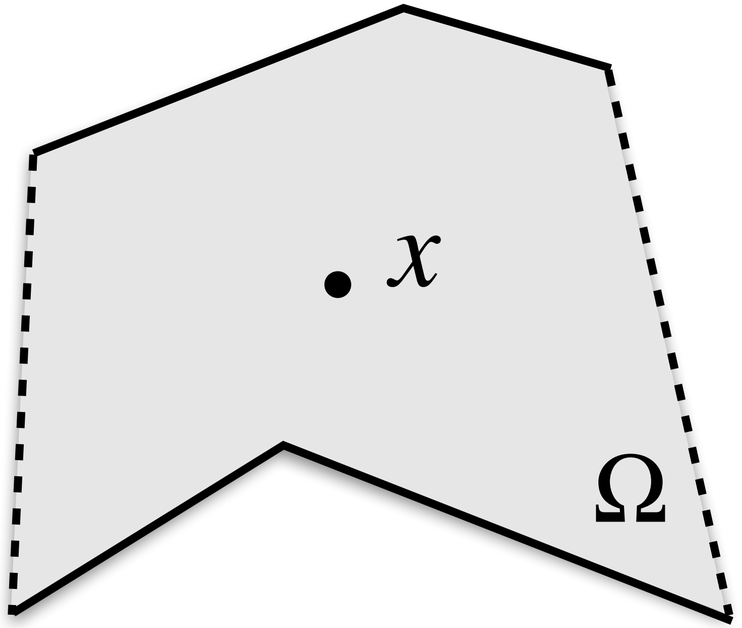
free-space Green's function

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

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

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

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



Boundary Integral Equation

free-space Poisson kernel

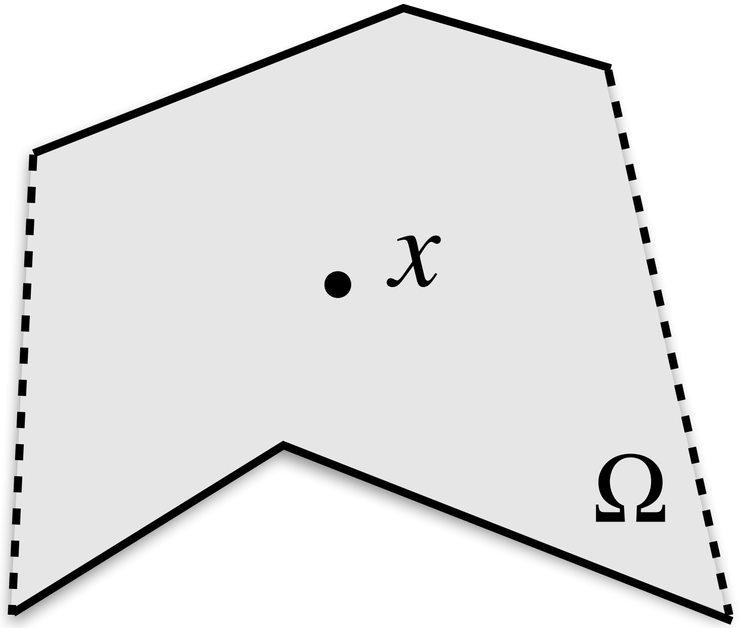
free-space Green's function

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

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

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

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



known

Boundary Integral Equation

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

free-space Poisson kernel

Dirichlet

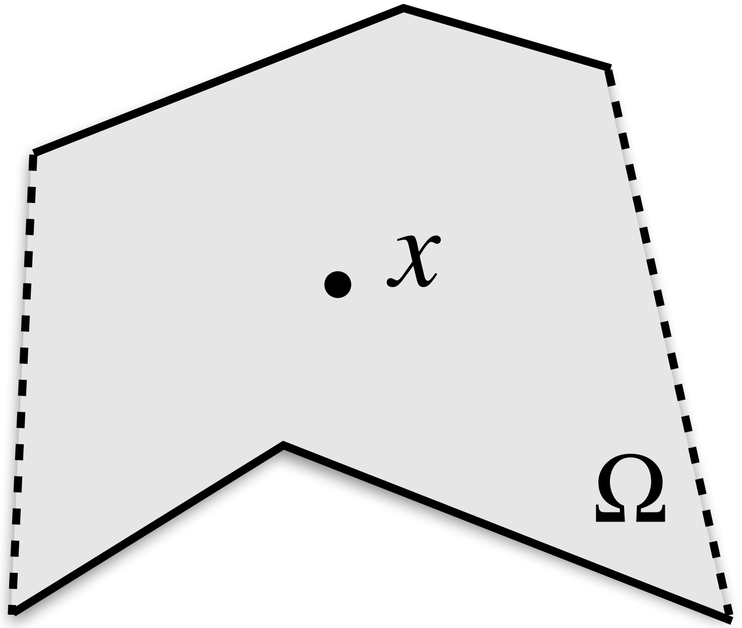
free-space Green's function

Neumann

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

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

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



known

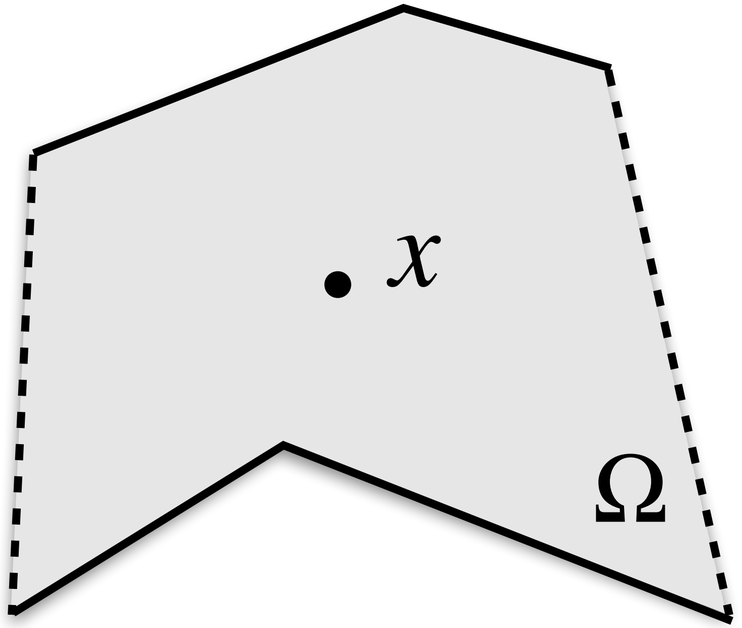
Monte Carlo estimator for BIE

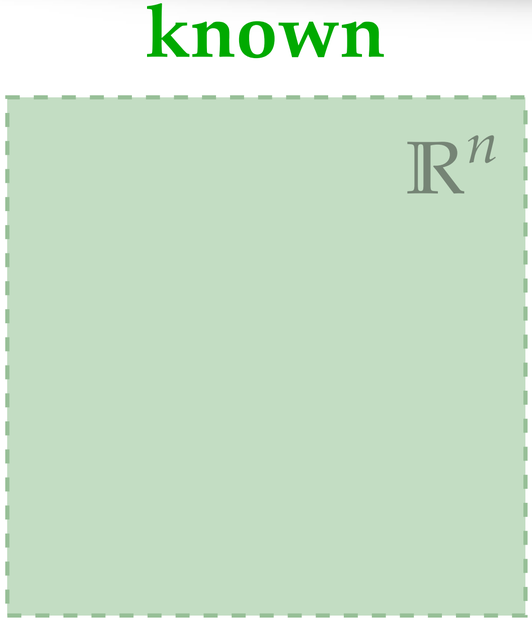
$$\widehat{u}(x) = \frac{|\partial\Omega|}{N} \sum_{i=1}^{N} \frac{\partial G(x, y_i)}{\partial n} u(y_i) - G(x, y_i) \frac{\partial u(y_i)}{\partial n}$$

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

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

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





Monte Carlo estimator for BIE

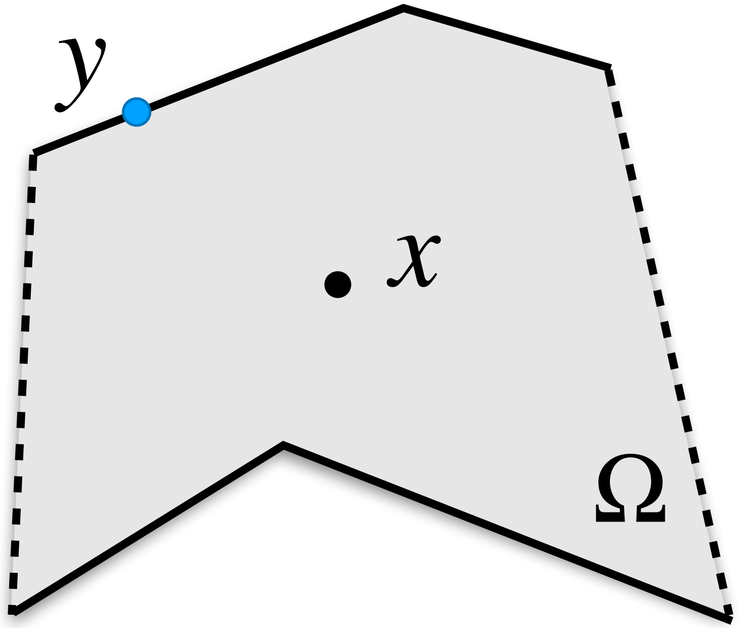
known estimate

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

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

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

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



known

Monte Carlo estimator for BIE

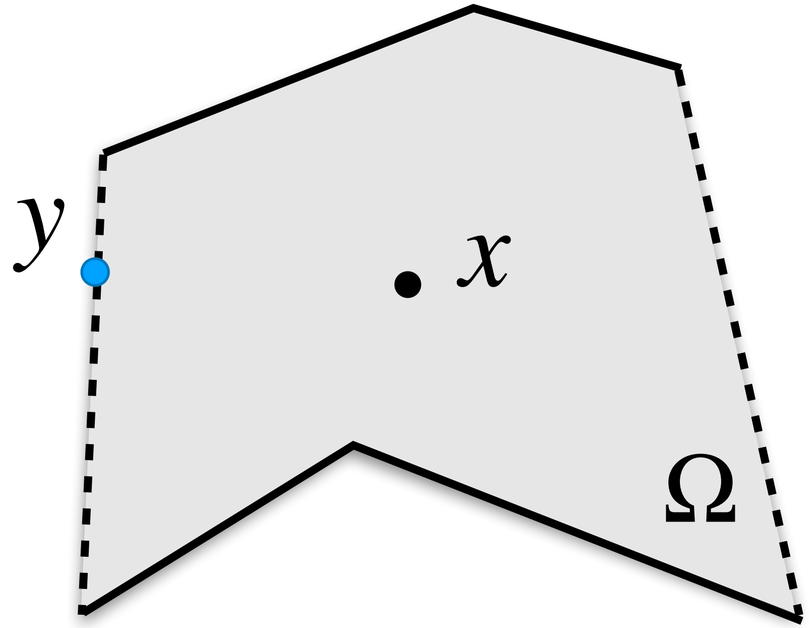
estimate known

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

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

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

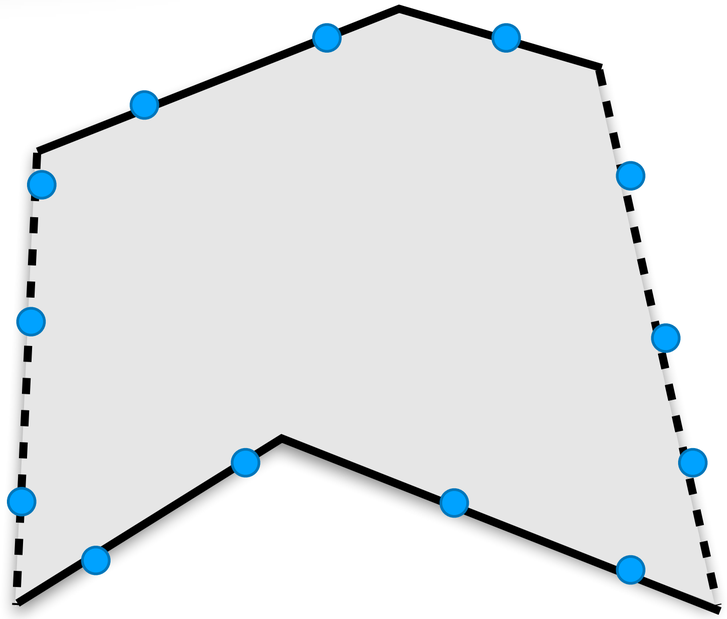
$$\frac{\partial u}{\partial n} = h \quad \text{on } \partial\Omega_N$$



known

Monte Carlo estimator for BIE

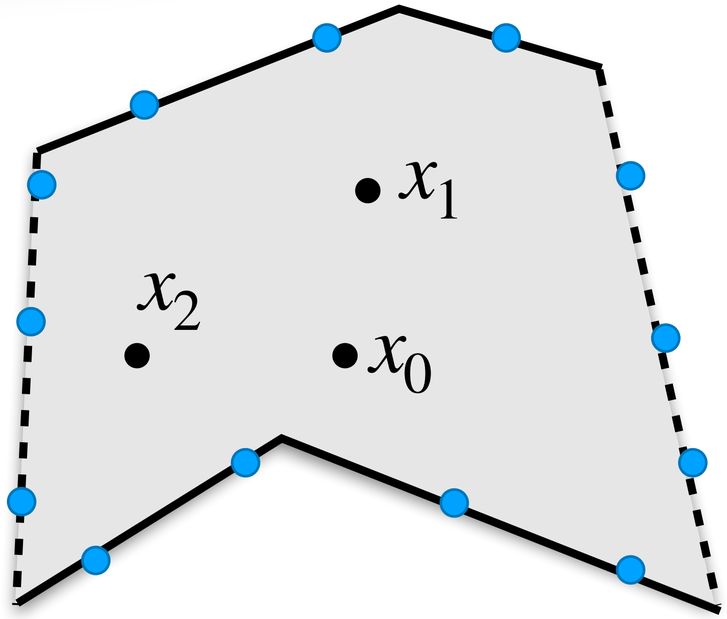
$$\hat{u}(x) = \frac{|\partial\Omega|}{N} \sum_{i=1}^{N} \frac{\partial G(x, y_i)}{\partial n} \hat{u}(y_i) - G(x, y_i) \widehat{\frac{\partial u(y_i)}{\partial n}}$$



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

Monte Carlo estimator for BIE

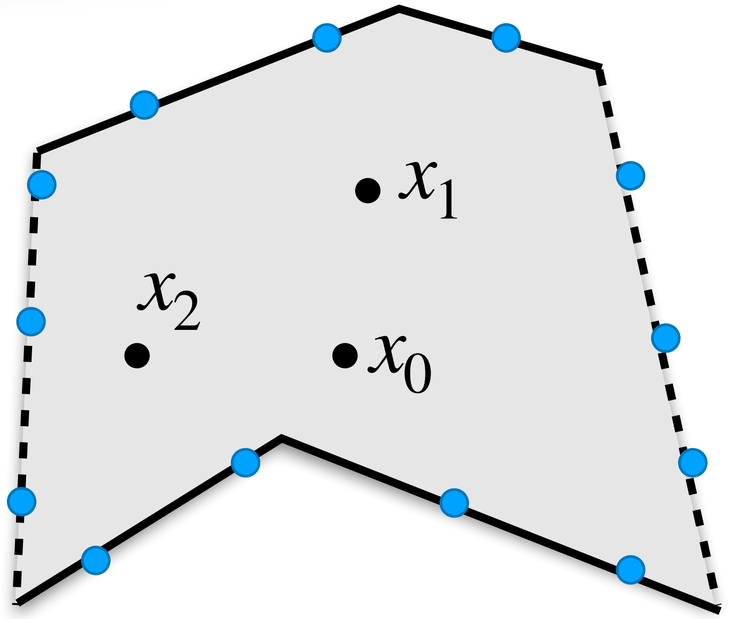
$$\widehat{u}(x) = \frac{|\partial\Omega|}{N} \sum_{i=1}^{N} \frac{\partial G(x, y_i)}{\partial n} \widehat{u}(y_i) - G(x, y_i) \widehat{\frac{\partial u(y_i)}{\partial n}}$$



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

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

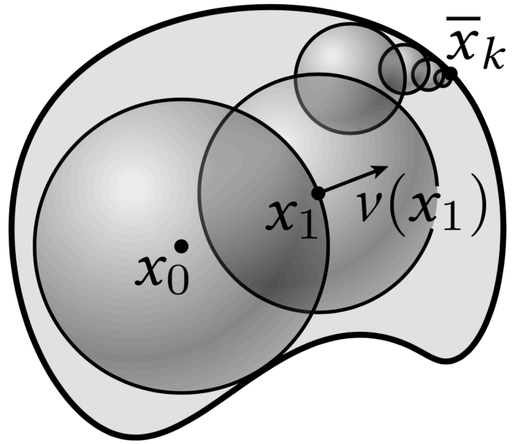
i.e., can estimate many integrals
with one set of estimates

Normal derivative estimation

Spatial derivative **inside ball**:

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

[Sawhney & Crane 2020]



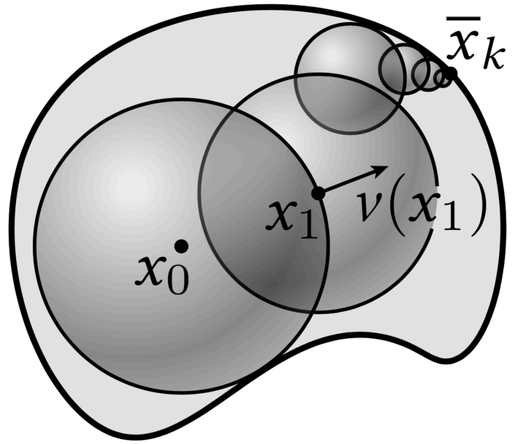
Normal derivative estimation

Spatial derivative **inside ball**:

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

normal

[Sawhney & Crane 2020]



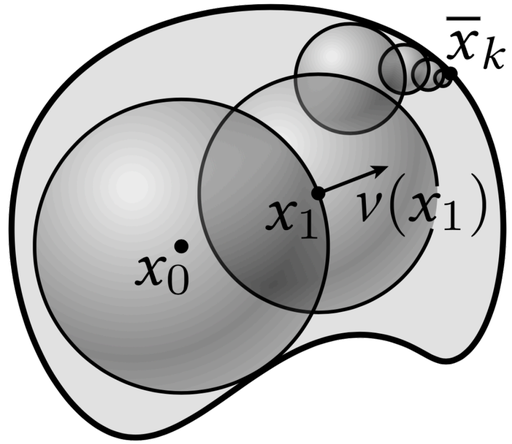
Normal derivative estimation

Spatial derivative **inside ball**:

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

$v(y)$: normal

[Sawhney & Crane 2020]



Normal derivative **on boundary**:

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

Normal derivative estimation

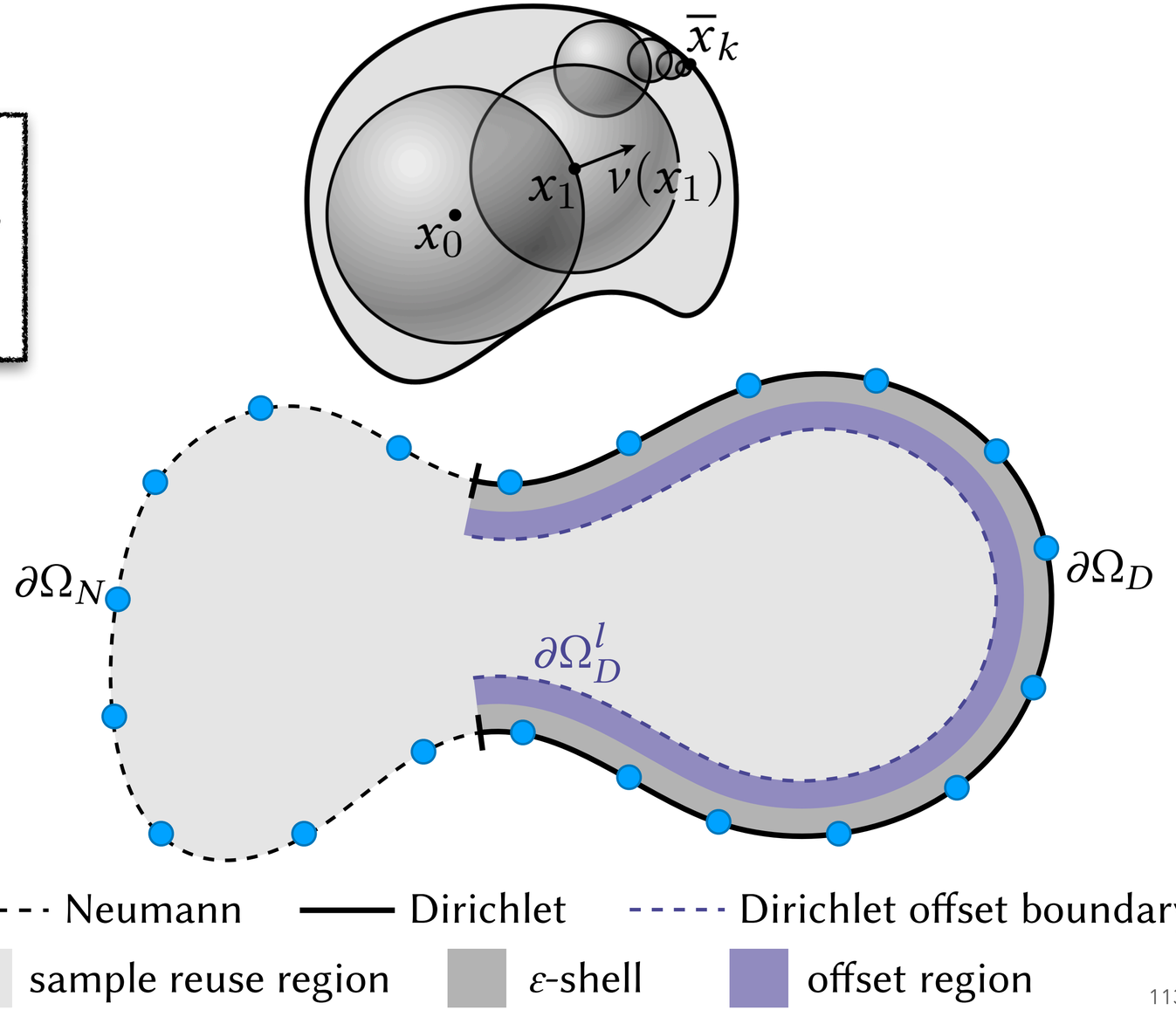
Spatial derivative **inside ball**:

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

[Sawhney & Crane 2020]

Normal derivative **on boundary**:

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



Normal derivative estimation

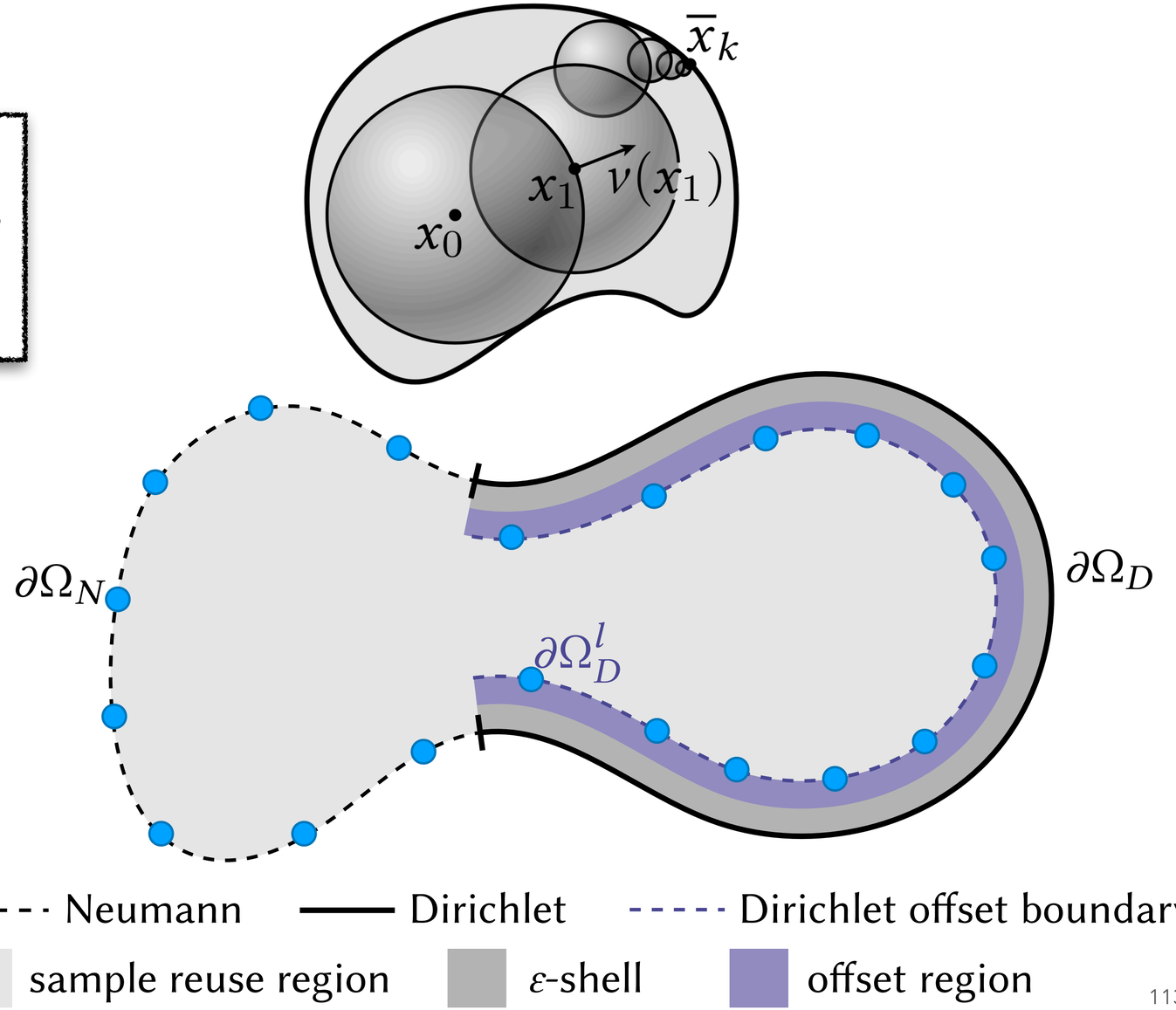
Spatial derivative **inside ball**:

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

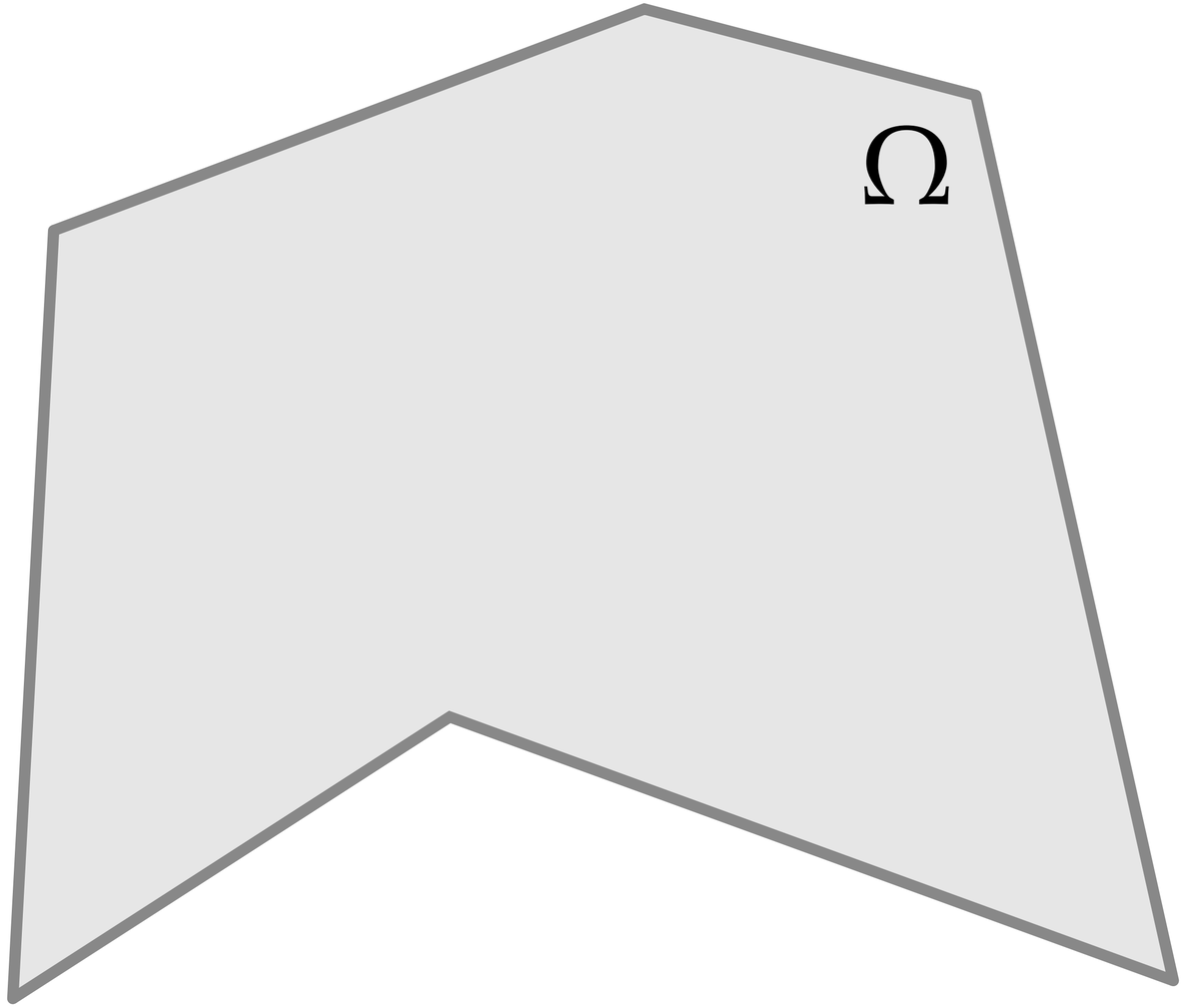
[Sawhney & Crane 2020]

Normal derivative **on boundary**:

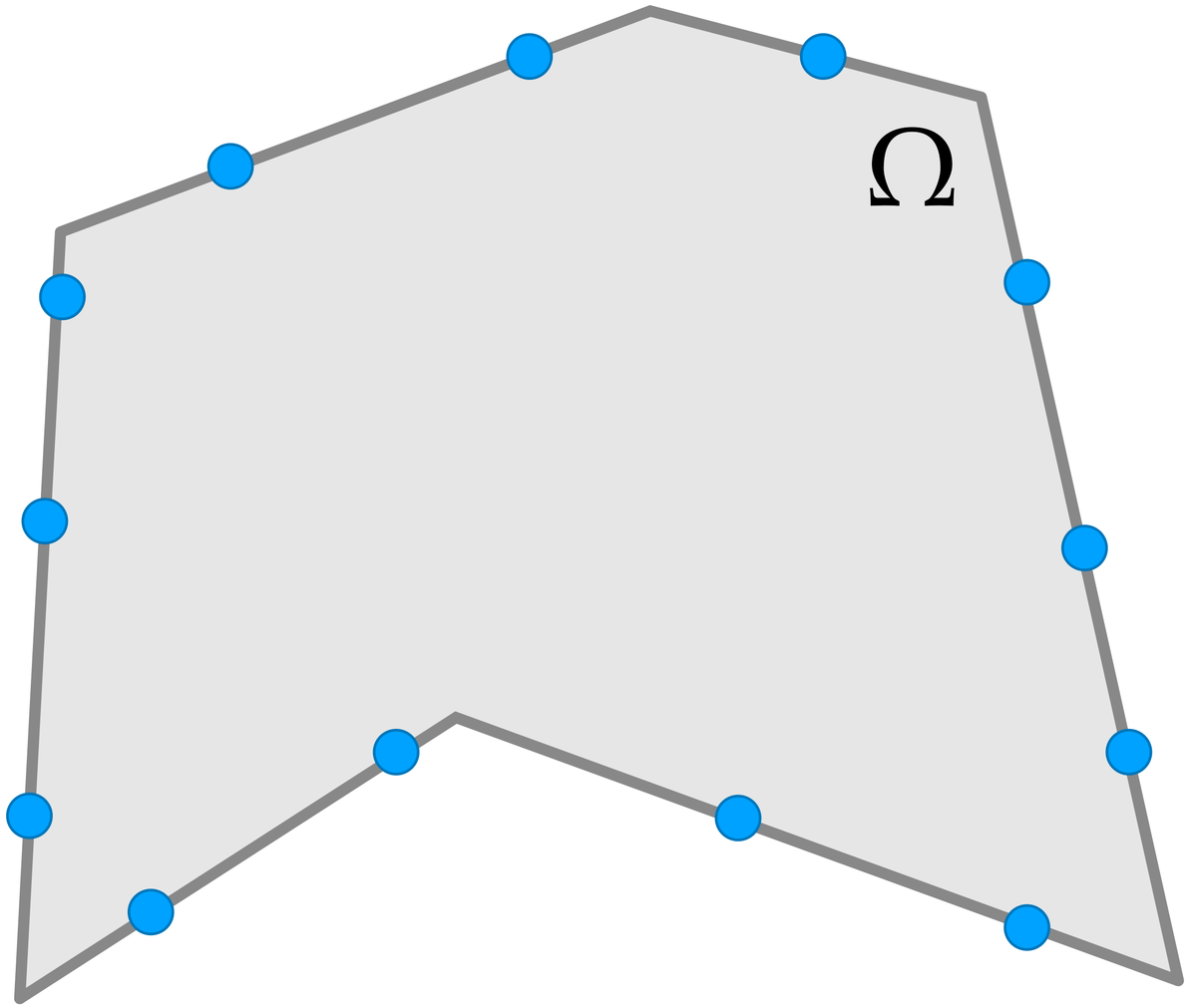
$$\frac{du(x)}{dn_x} = n_x \cdot \nabla_x u(x)$$



Boundary Value Caching (BVC)

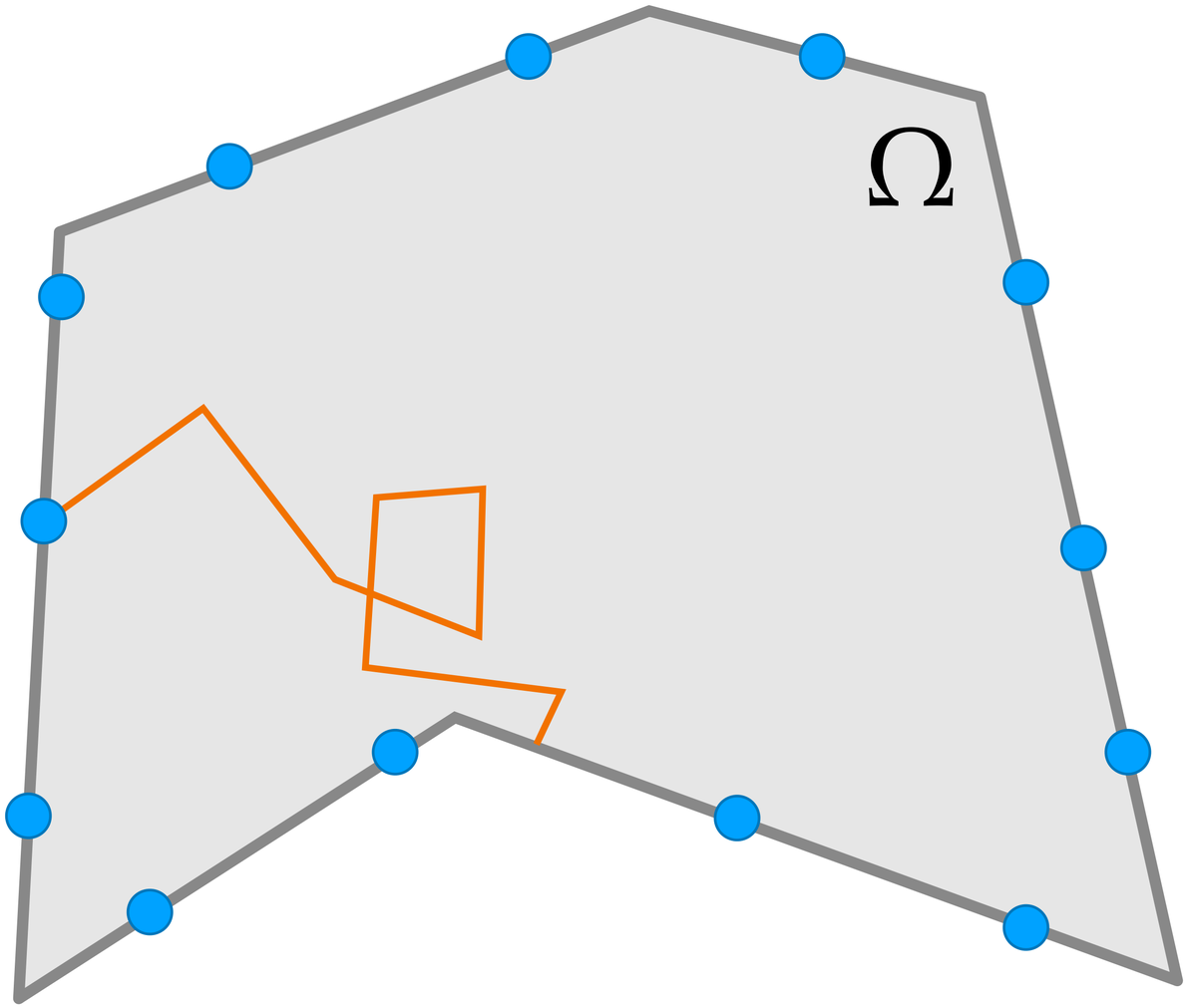


Boundary Value Caching (BVC)



generate samples on boundary $\partial\Omega$

Boundary Value Caching (BVC)

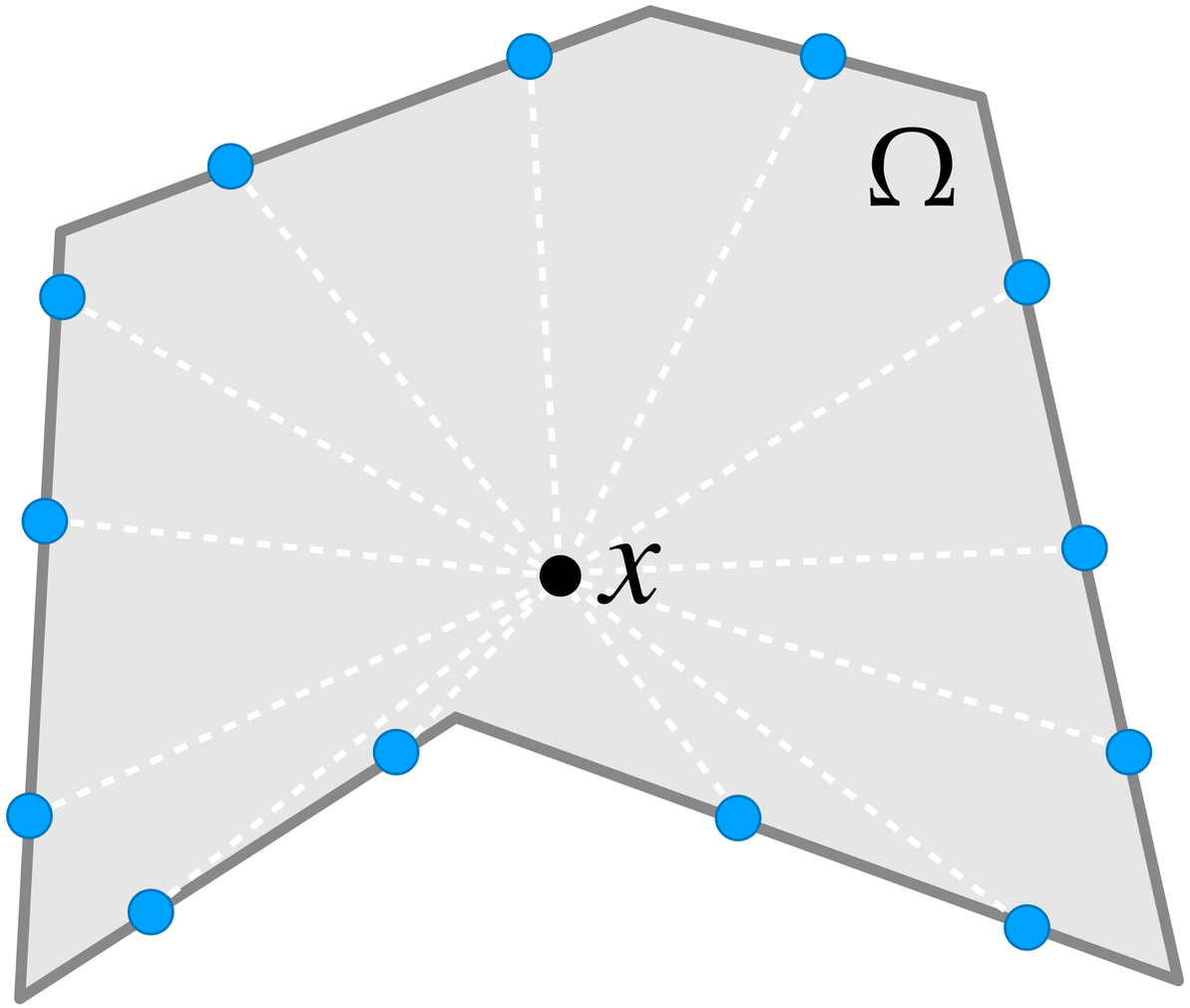


generate samples on boundary $\partial\Omega$

↓

use WoSt to estimate u & $\frac{du}{dn}$

Boundary Value Caching (BVC)



generate samples on boundary $\partial\Omega$

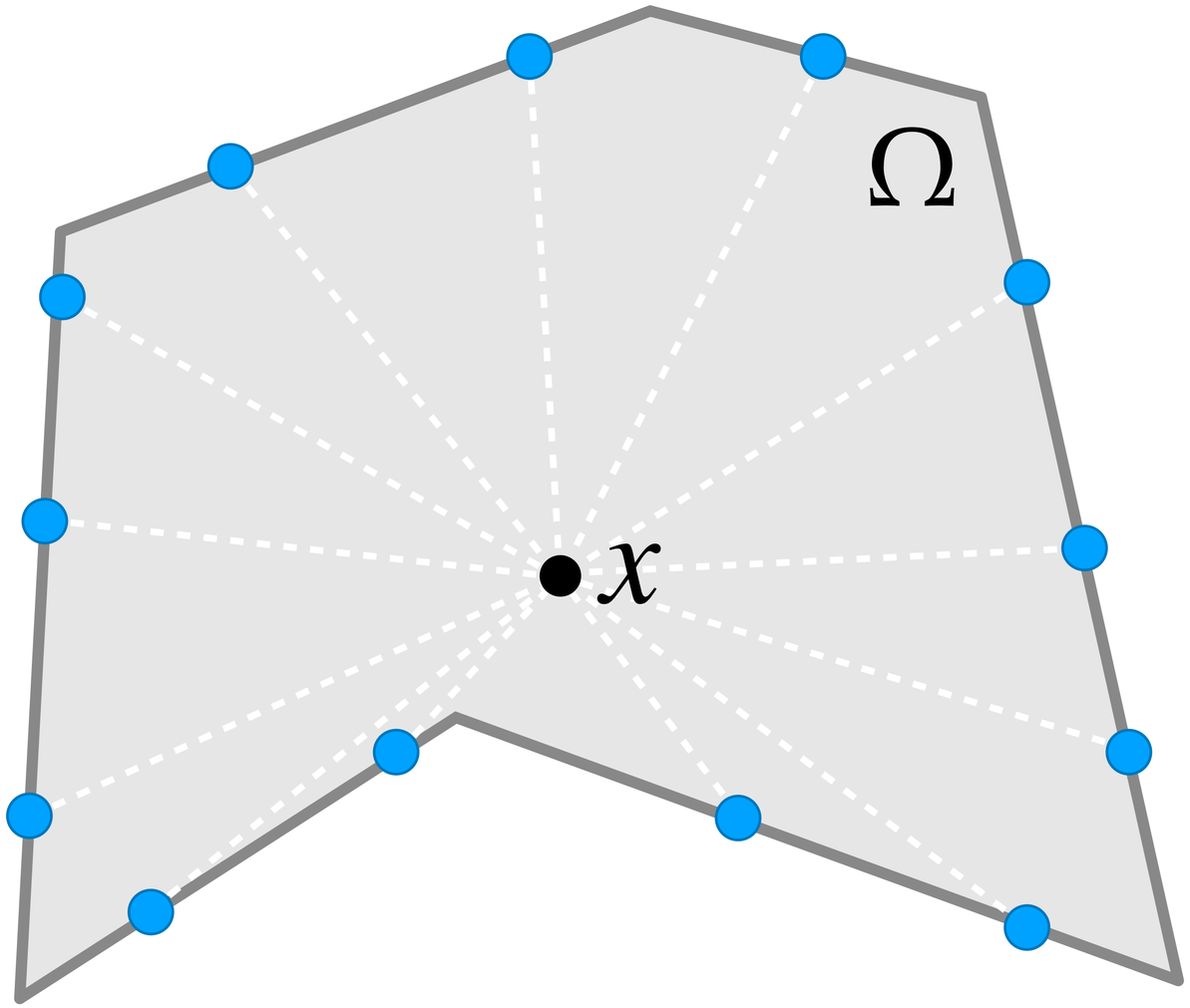
↓

use WoSt to estimate u & $\frac{du}{dn}$

↓

evaluate BIE inside domain Ω

Boundary Value Caching (BVC)

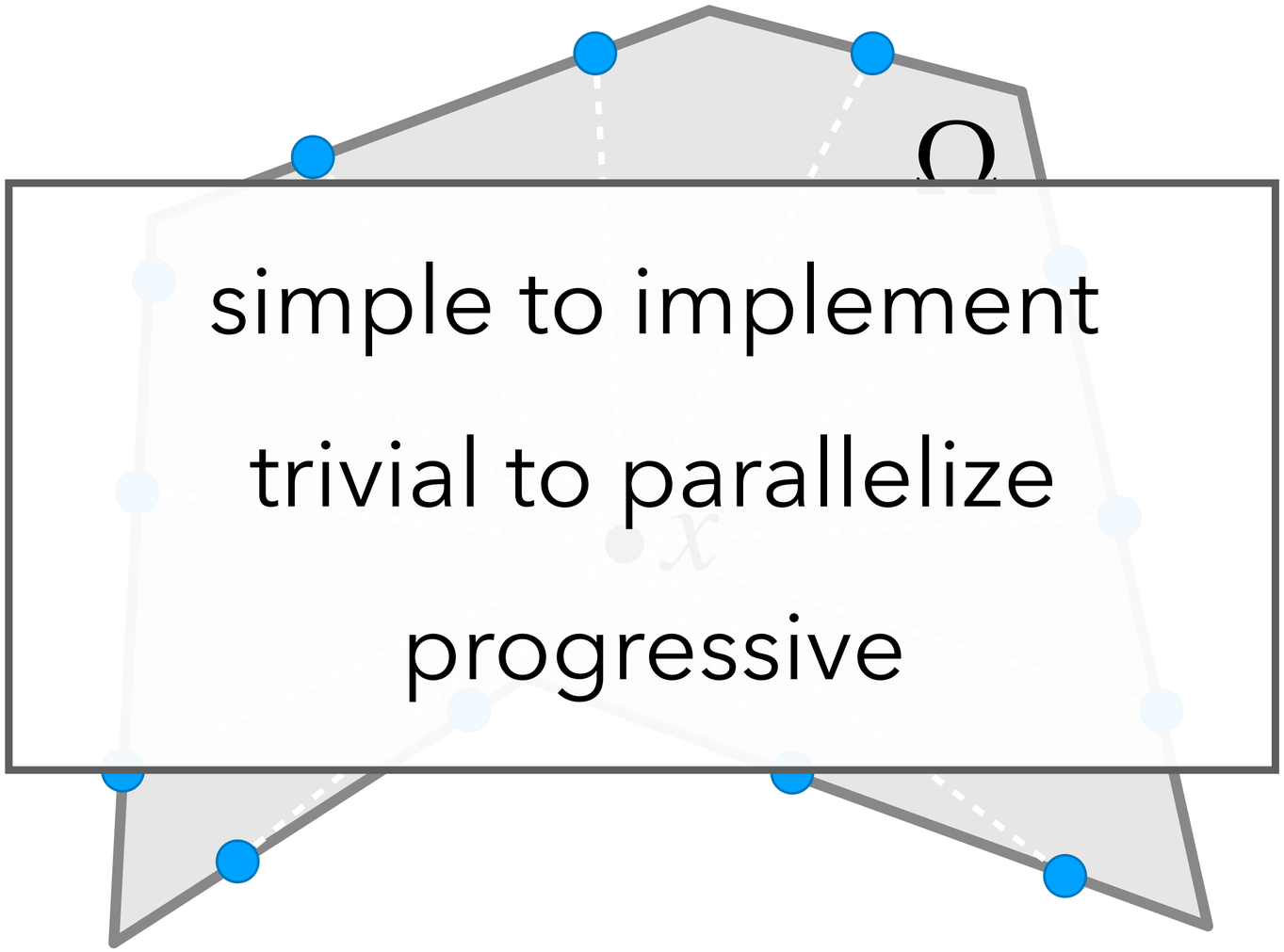


generate samples on boundary $\partial\Omega$

use WoSt to estimate u & $\frac{du}{dn}$

evaluate BIE inside domain Ω

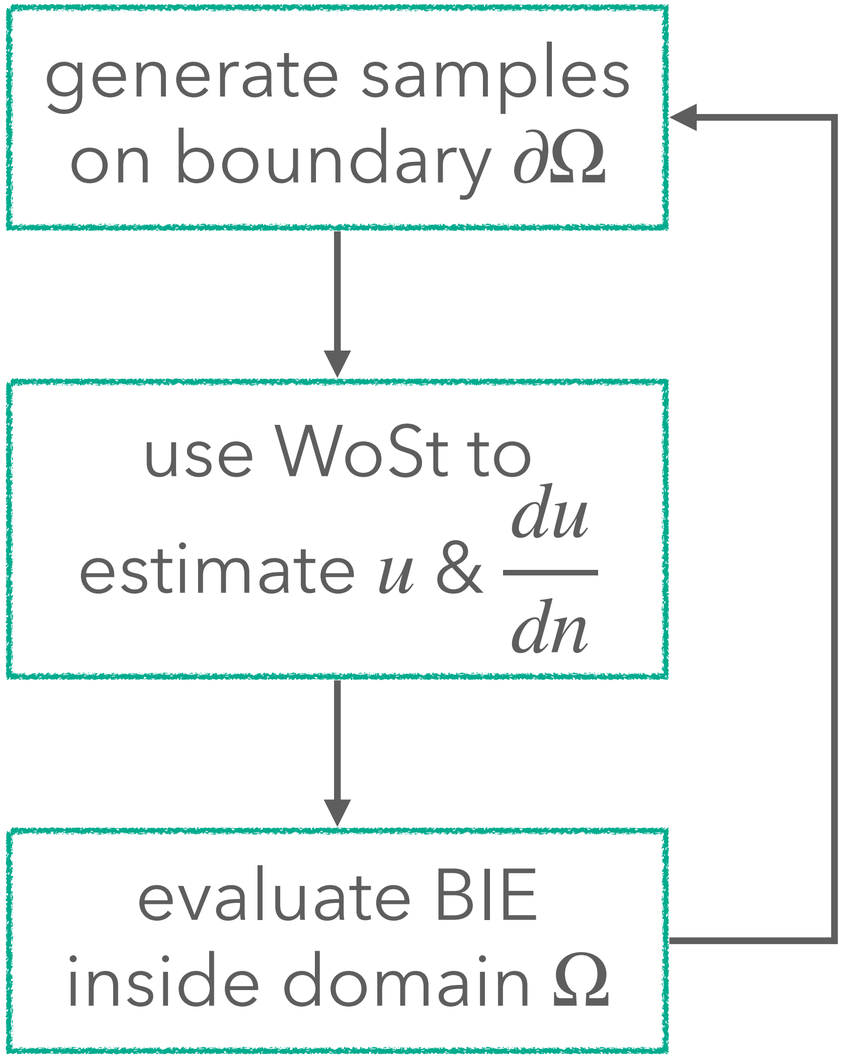
Boundary Value Caching (BVC)



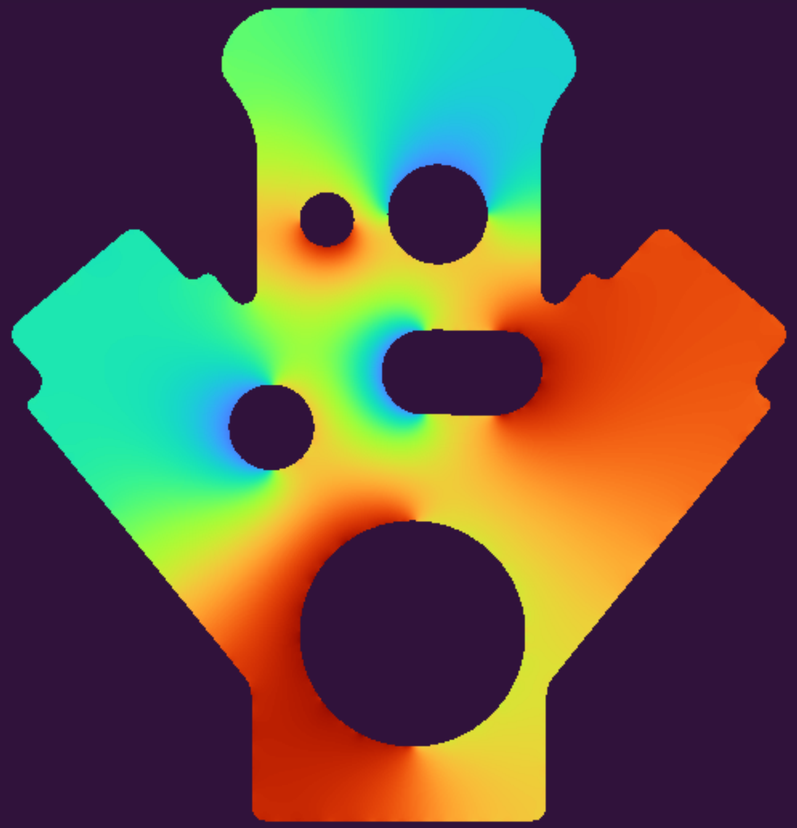
simple to implement

trivial to parallelize

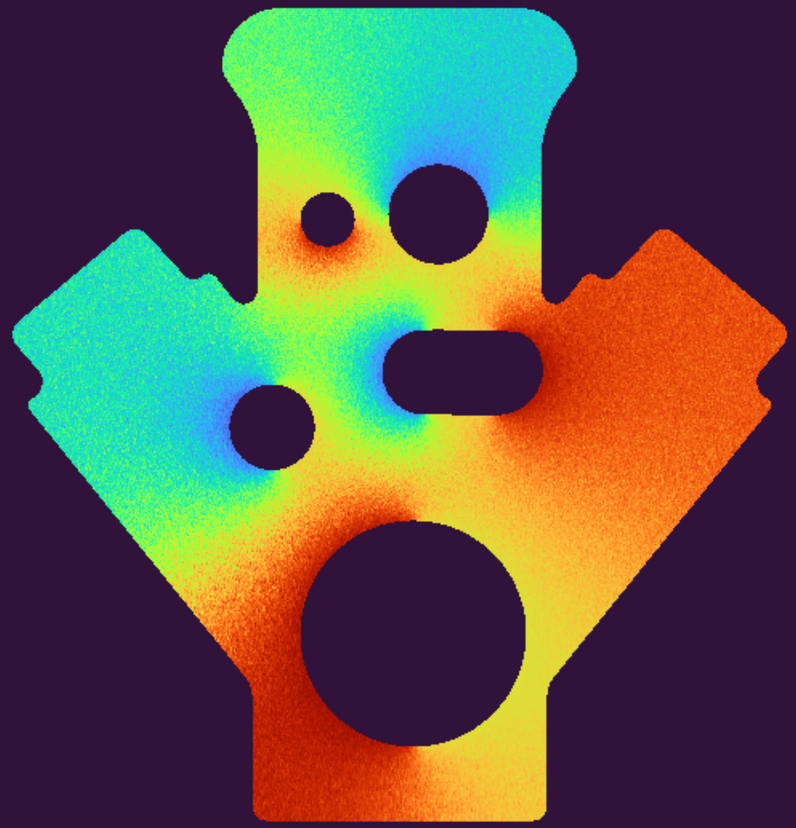
progressive



Benefits of BVC



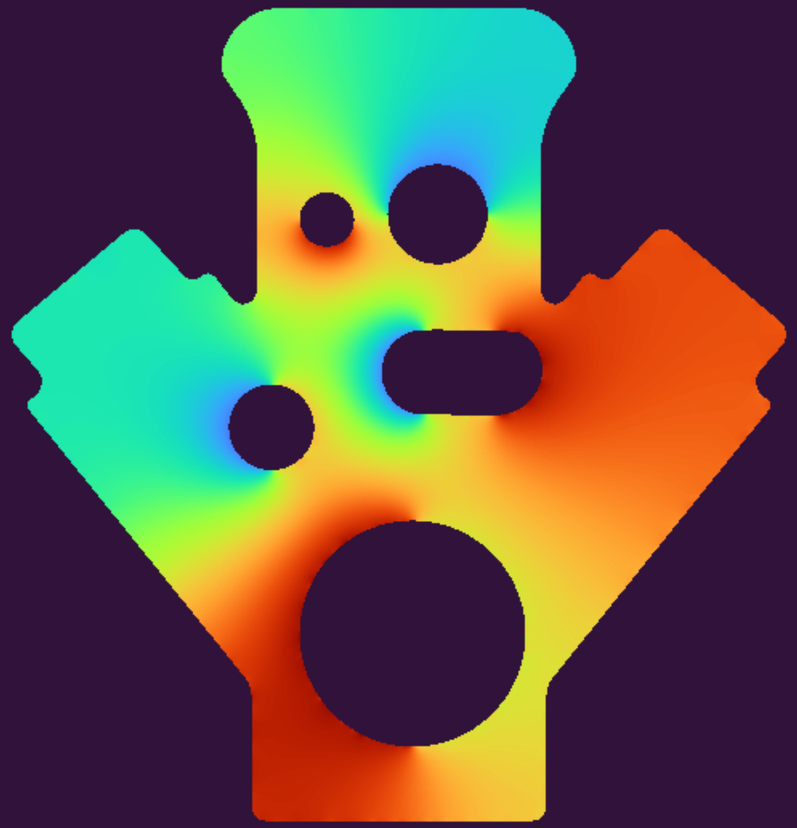
boundary value caching



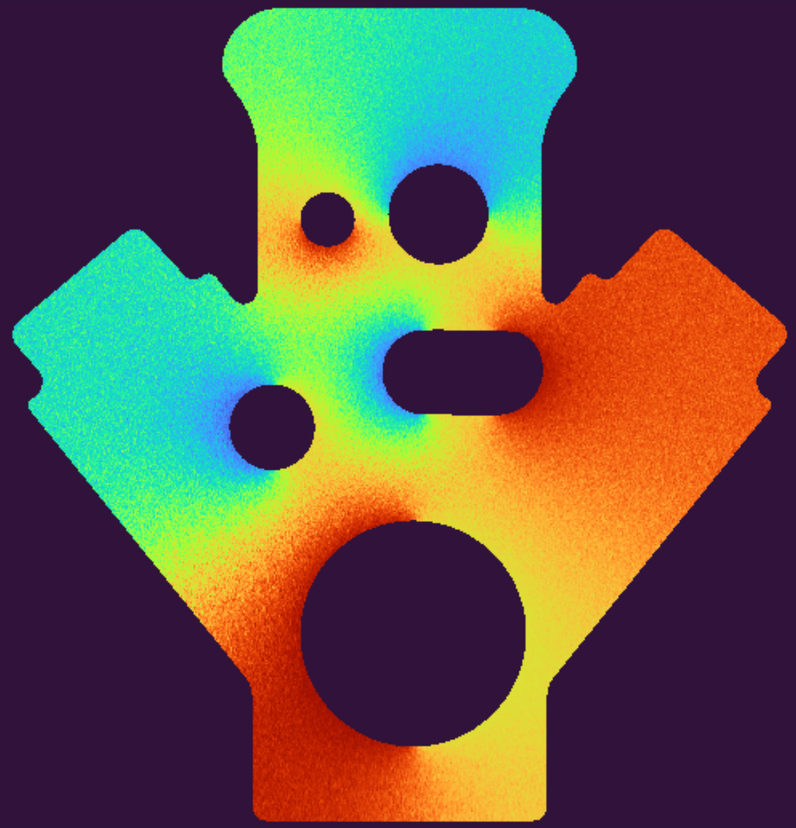
walk on stars

Improved run-time efficiency (sharing global information)

Benefits of BVC



boundary value caching



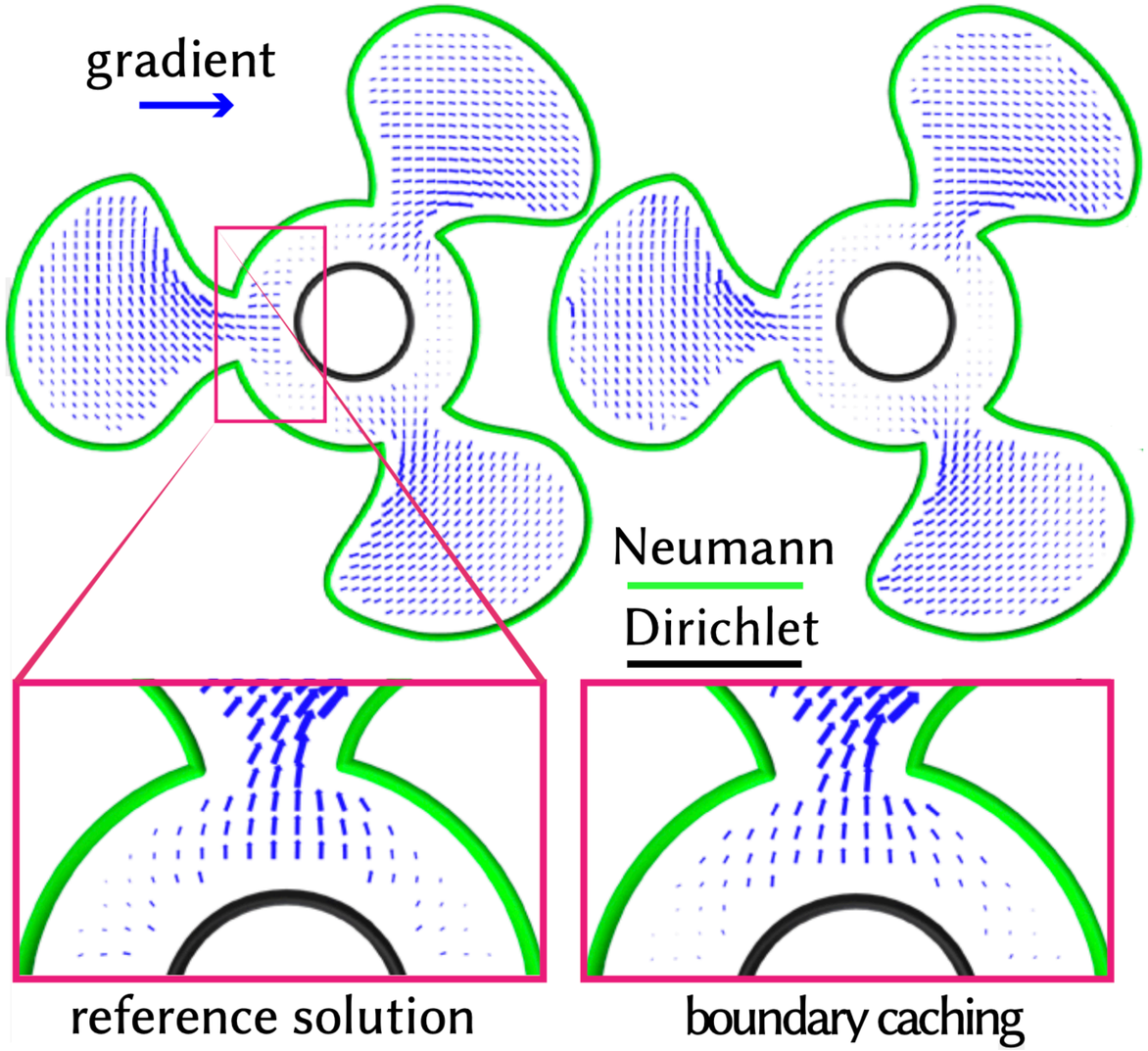
walk on stars

Improved run-time efficiency (sharing global information)

Suppressed noise (due to correlation)

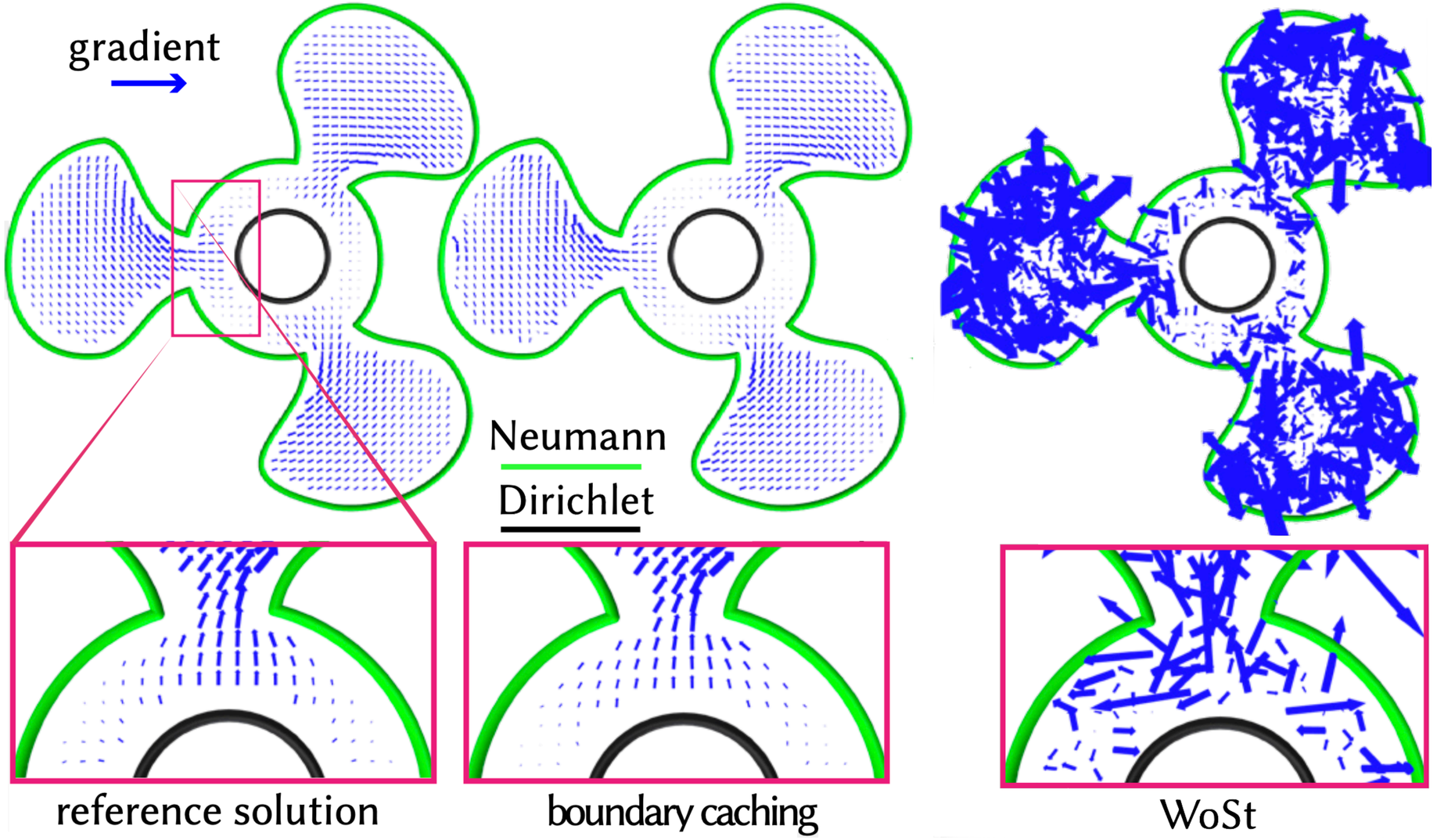
Gradient estimates with BVC

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



Gradient estimates with BVC

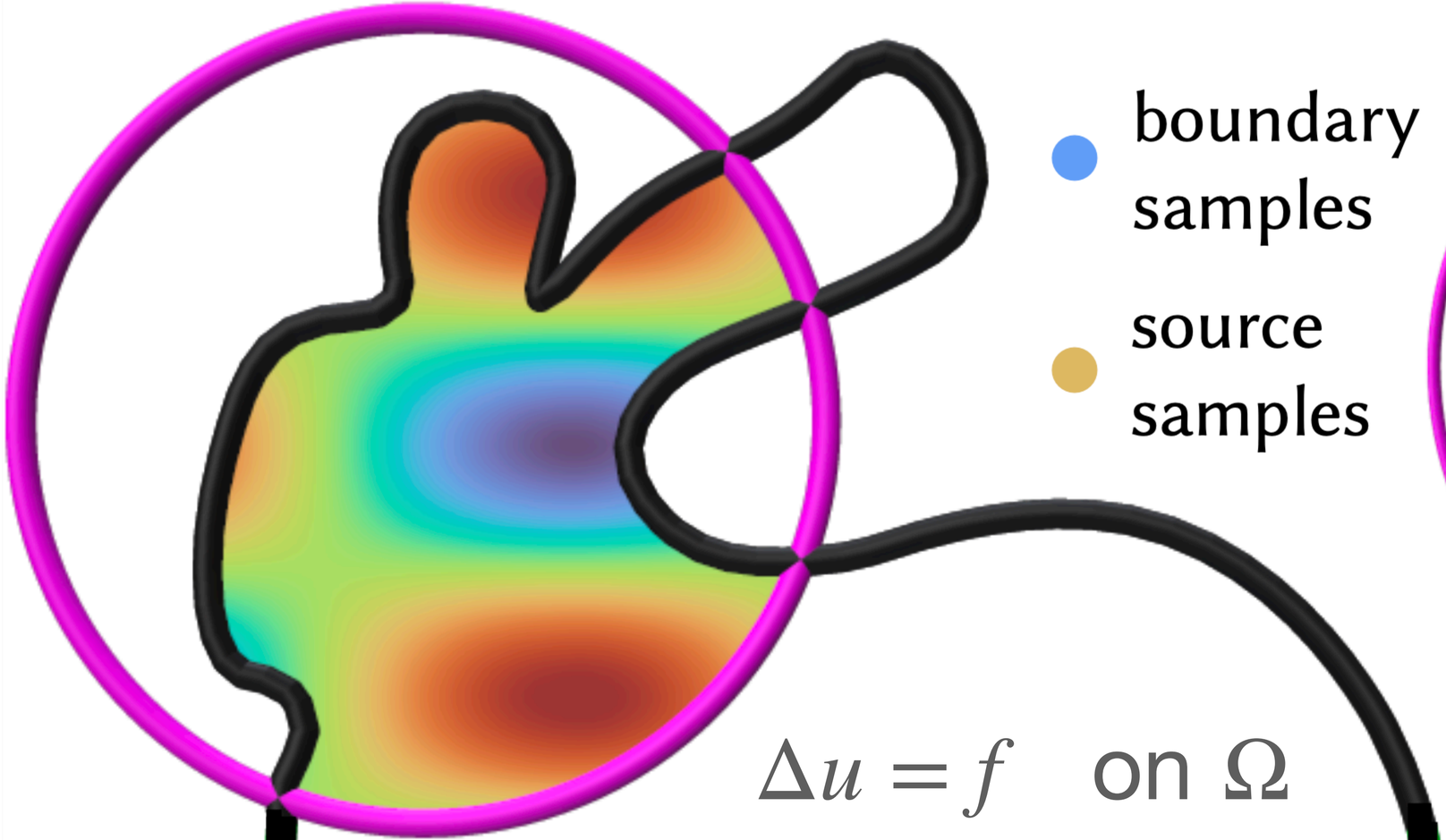
Can reuse **same** boundary cache for gradients!



Source term

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

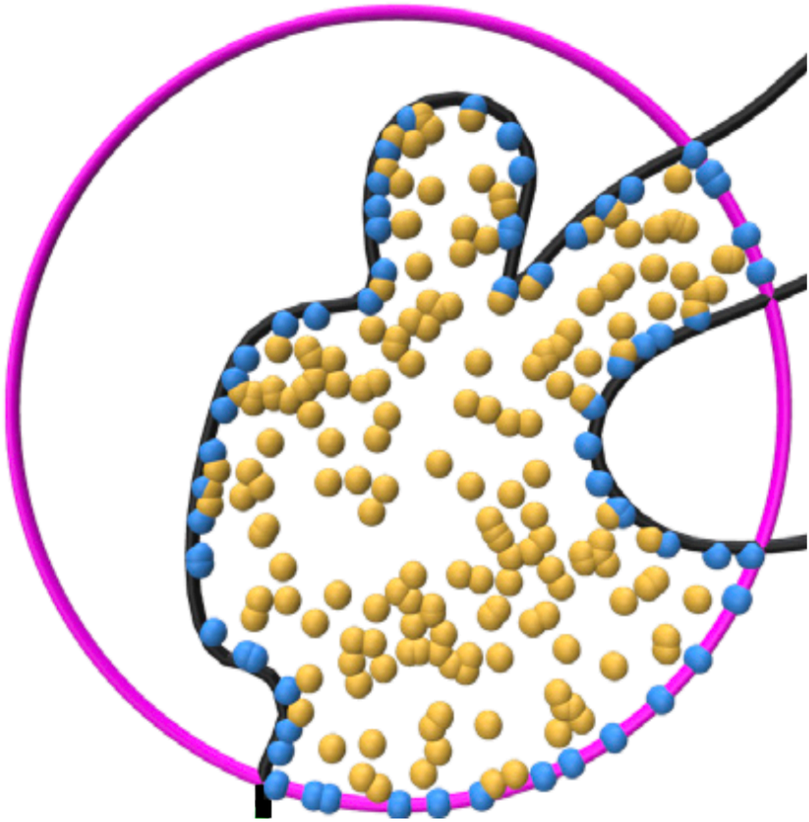
no random walks needed



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

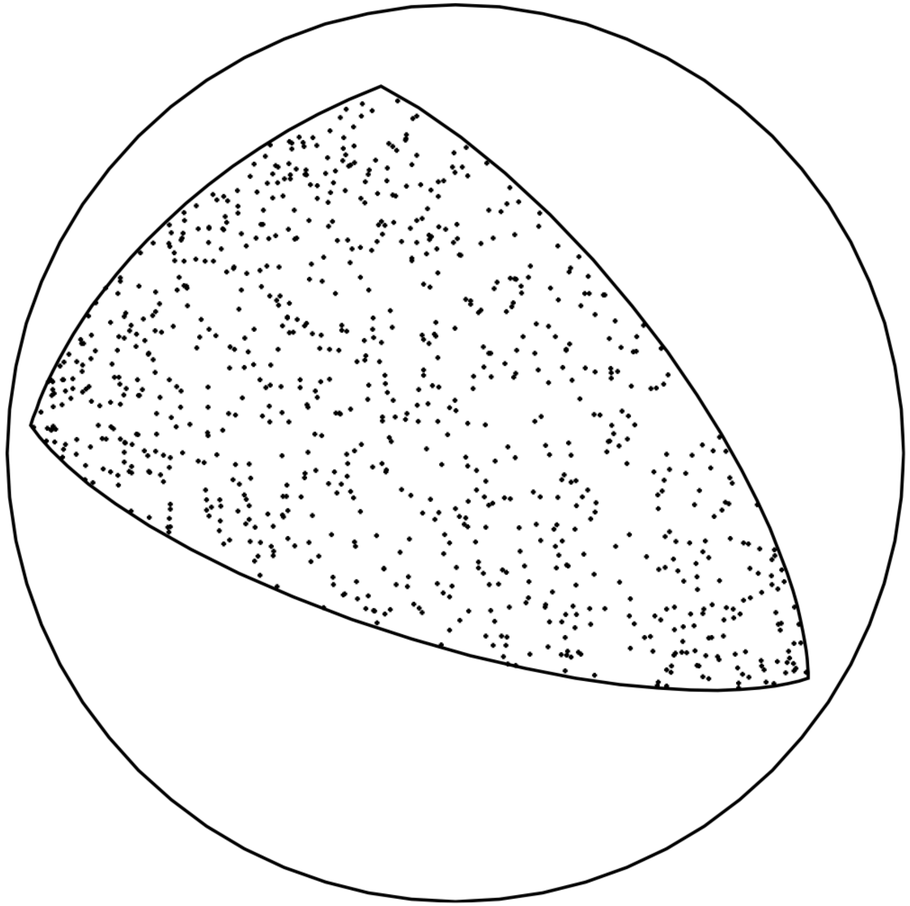
boundary samples

source samples



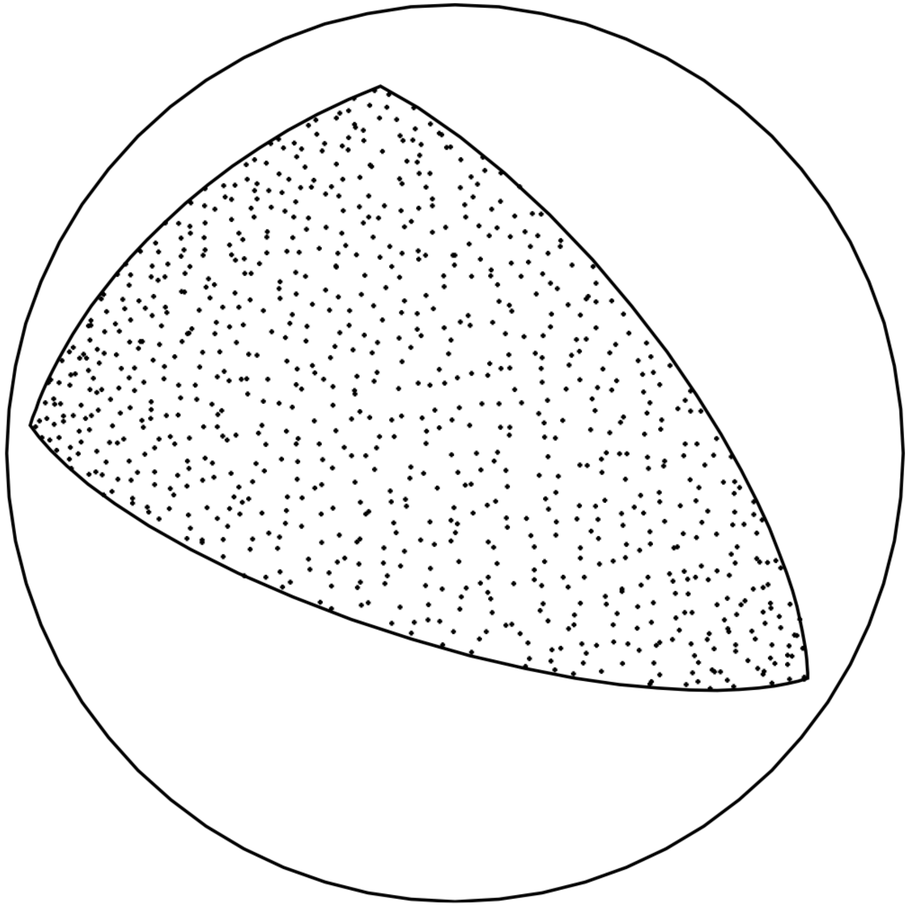
Stratification

uniform



“more clumpy”

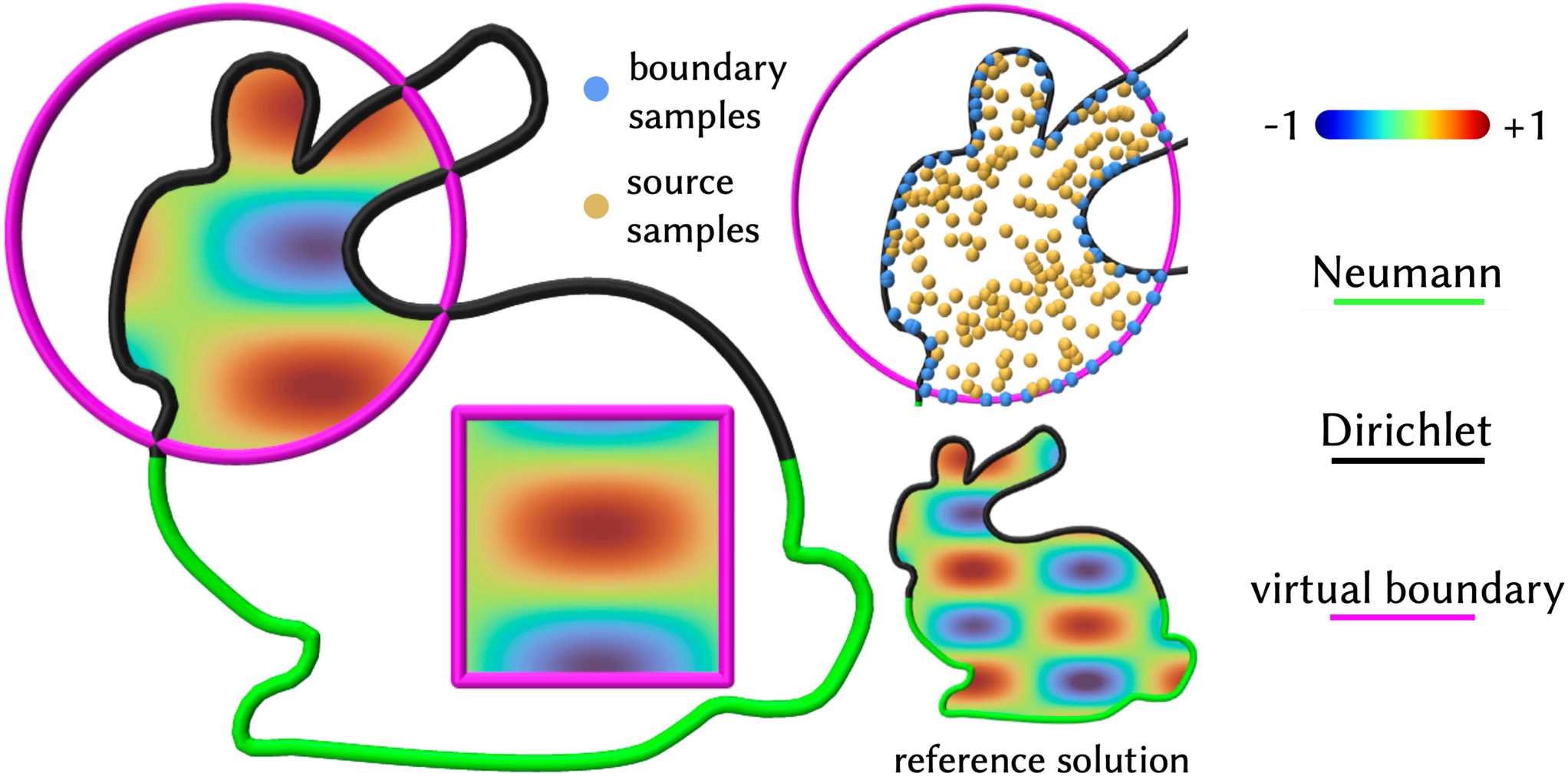
stratified



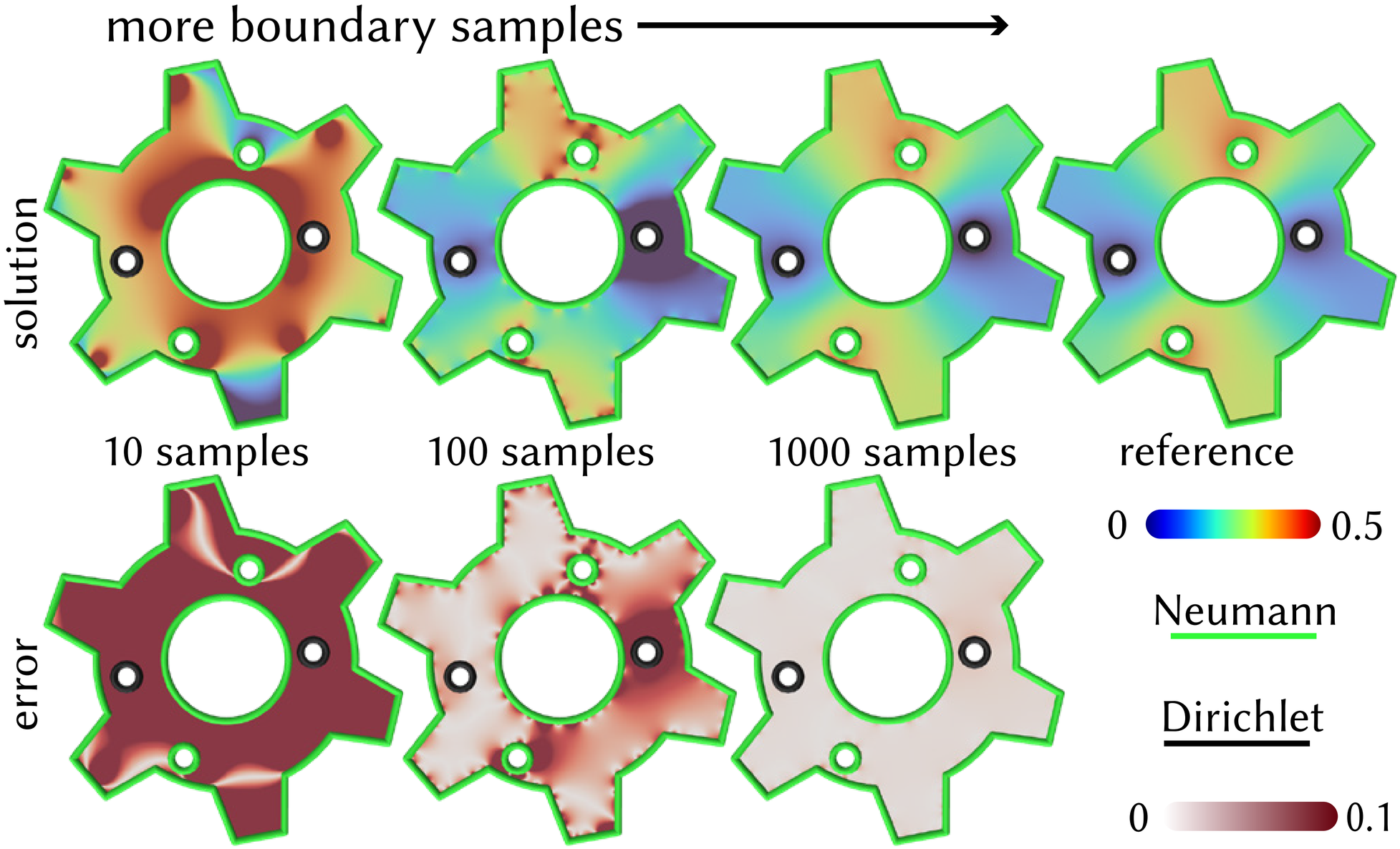
“more even”

Output sensitivity with BVC

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

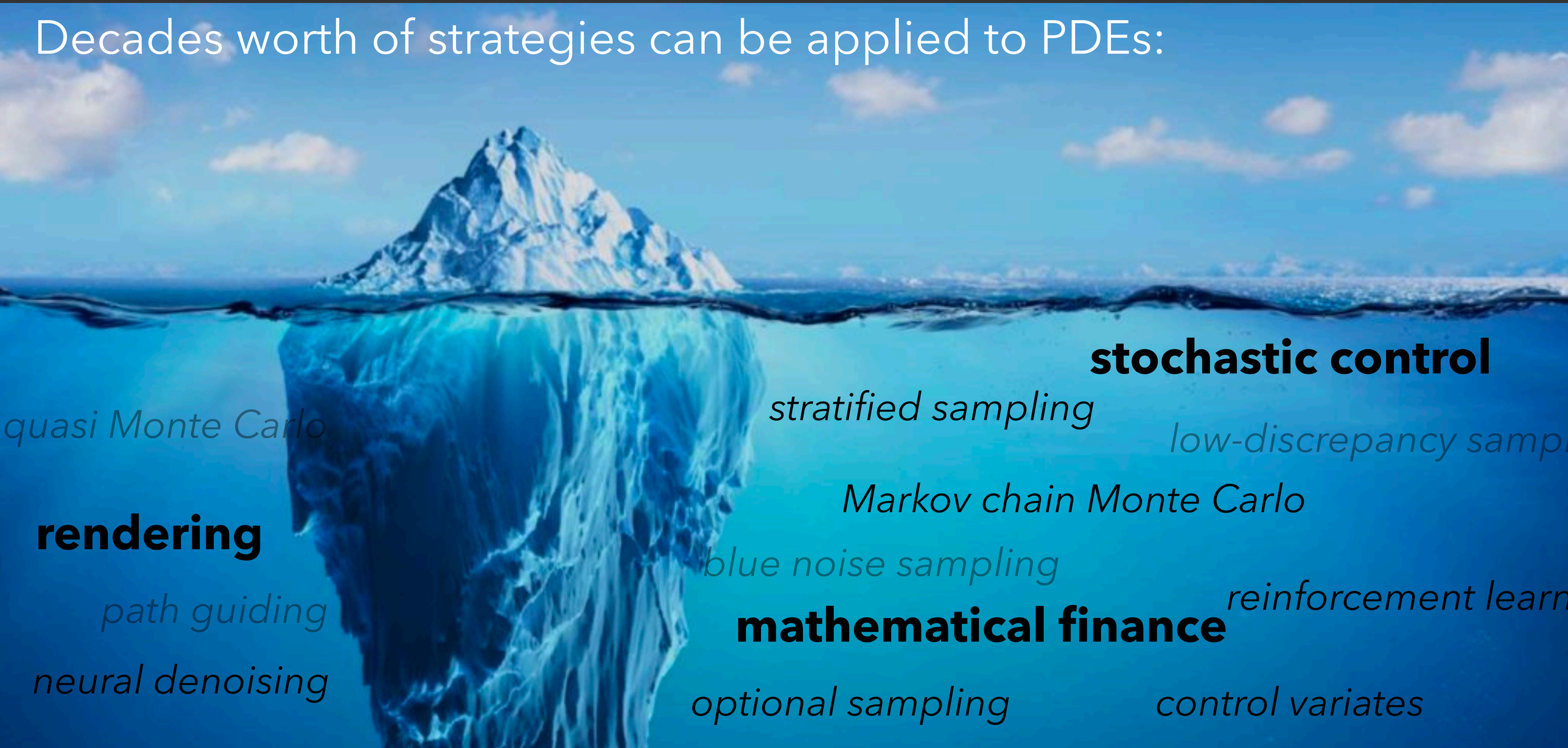


Error and convergence



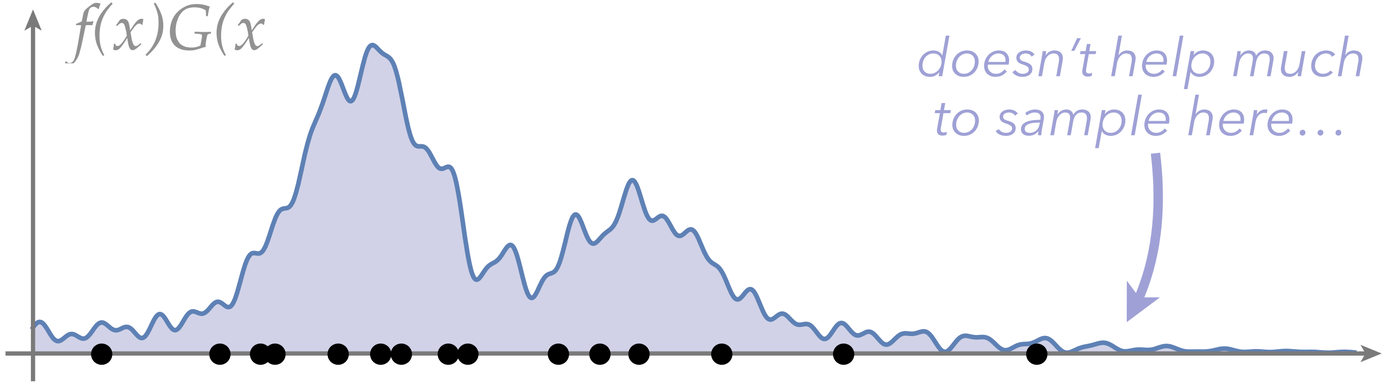
Just the tip of the iceberg…

Decades worth of strategies can be applied to PDEs:



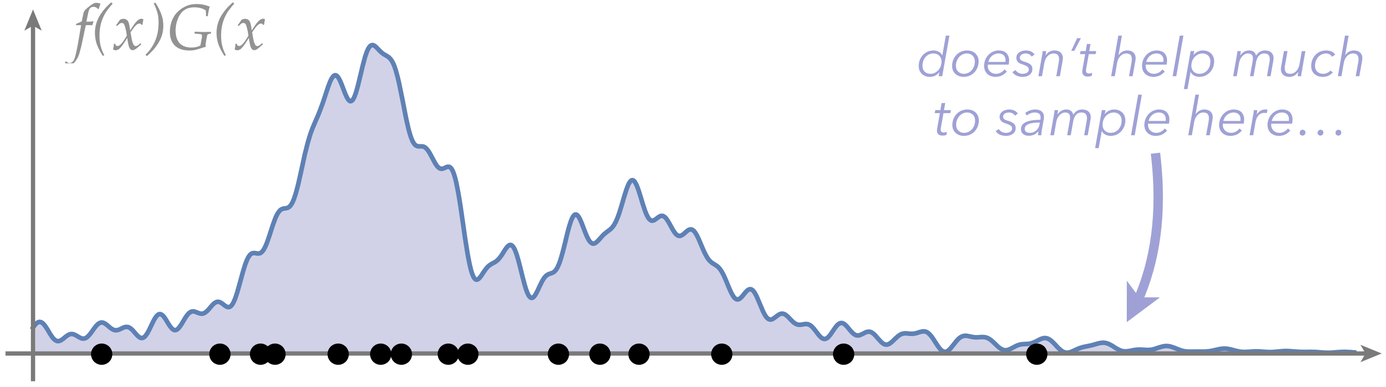
Importance sampling of Green's function & source term

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

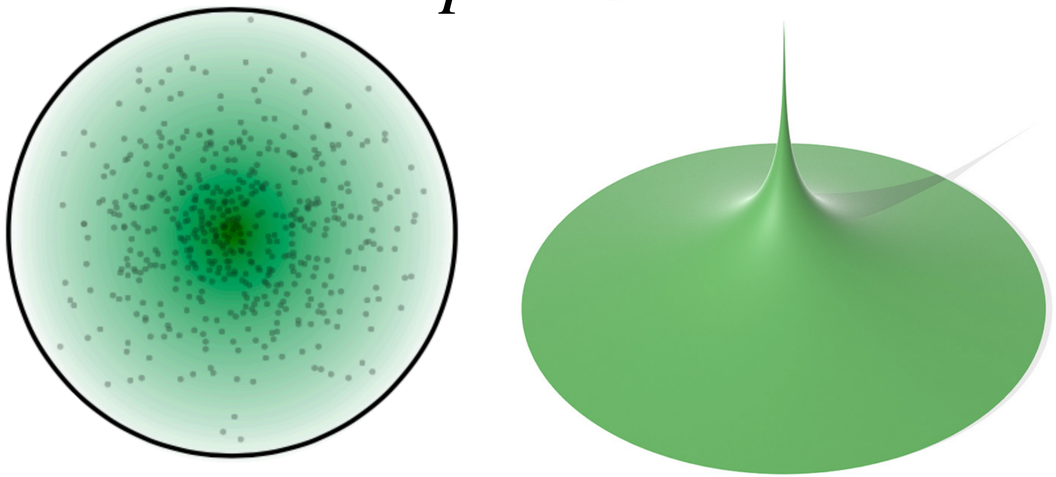


Importance sampling of Green's function & source term

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



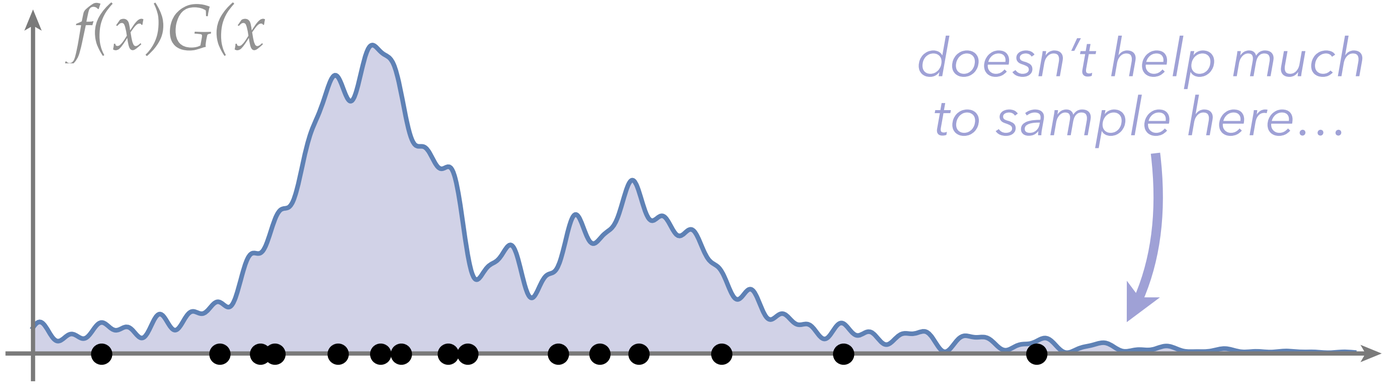
$$p \propto G$$



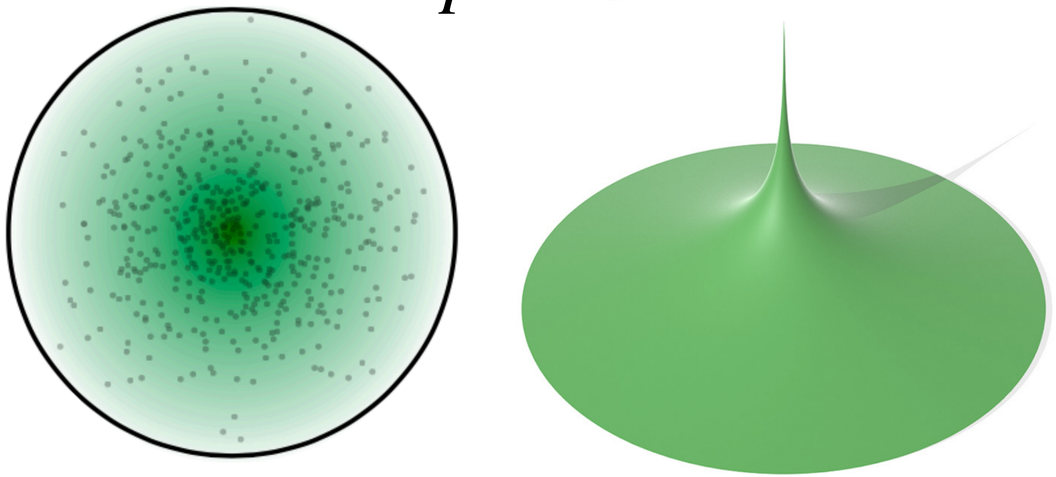
Samples drawn from Green's function of ball

Importance sampling of Green's function & source term

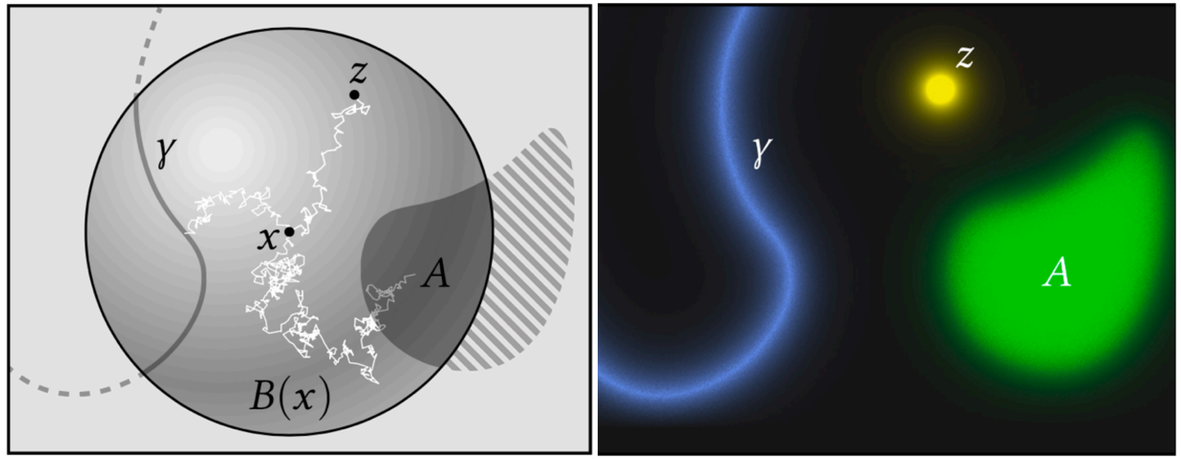
$$\hat{I} = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)\ G(X_i)}{p(X_i)}$$



$p \propto G$



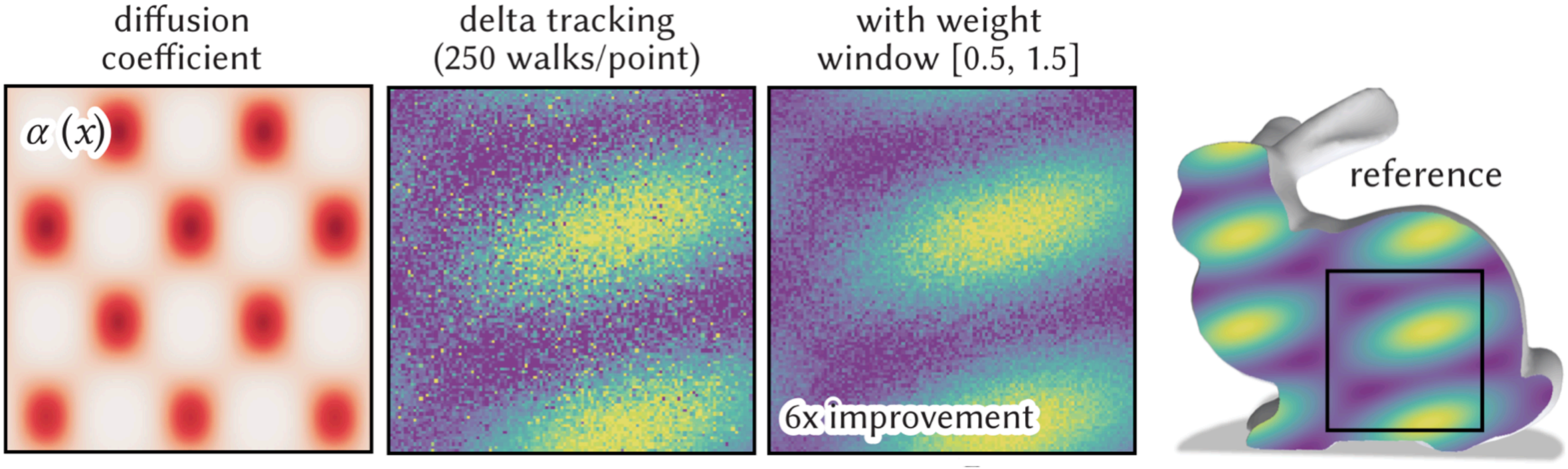
$p \propto f$



Samples drawn from Green's function of ball

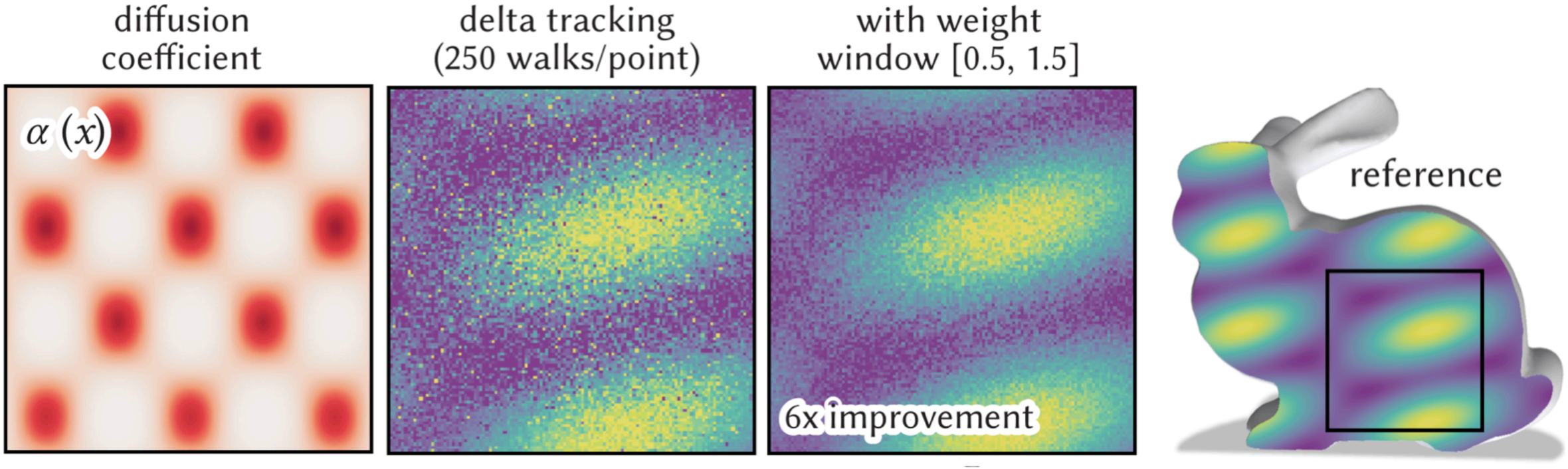
Sampling point, curve & area sources

Weight window for variable coefficient PDEs



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

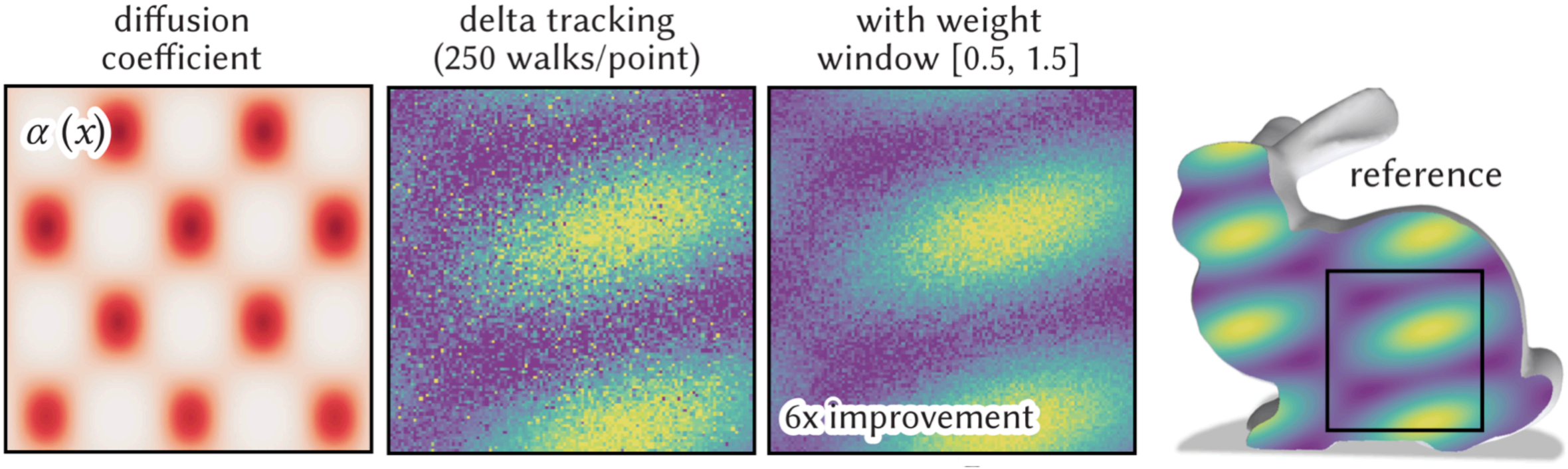
Weight window for variable coefficient PDEs



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

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

Weight window for variable coefficient PDEs



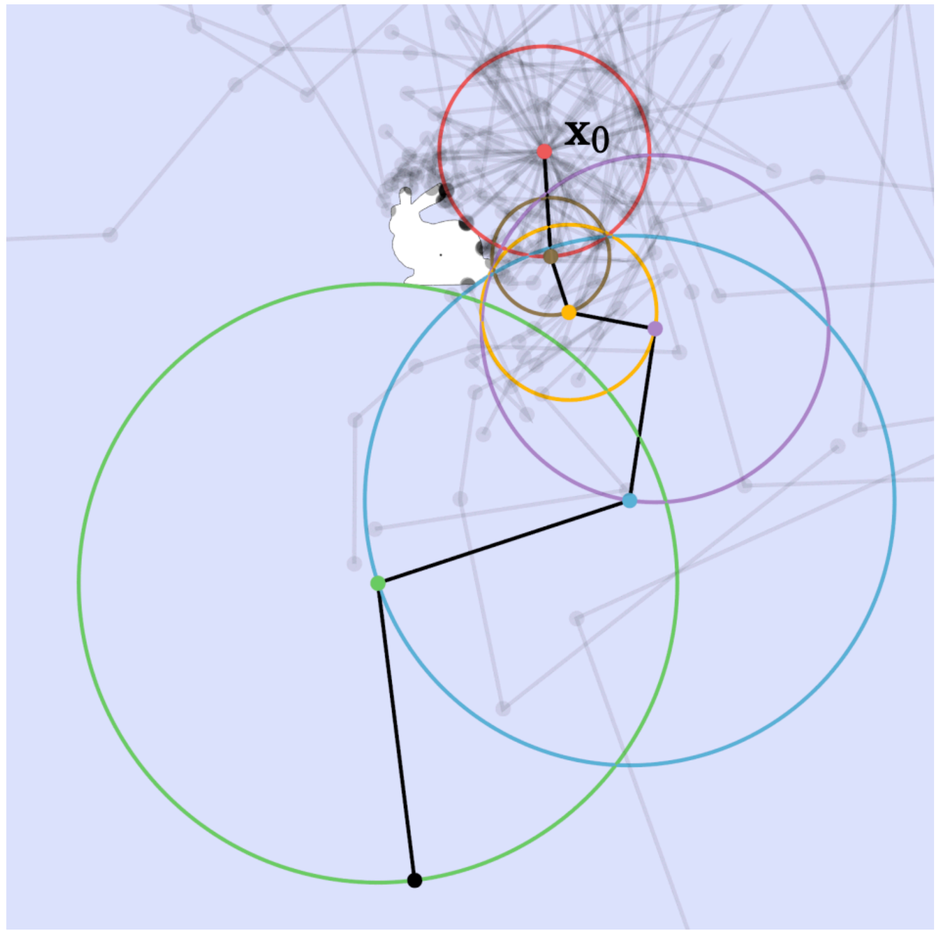
Probabilistically **terminate** low-contribution random walks to improve efficiency

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

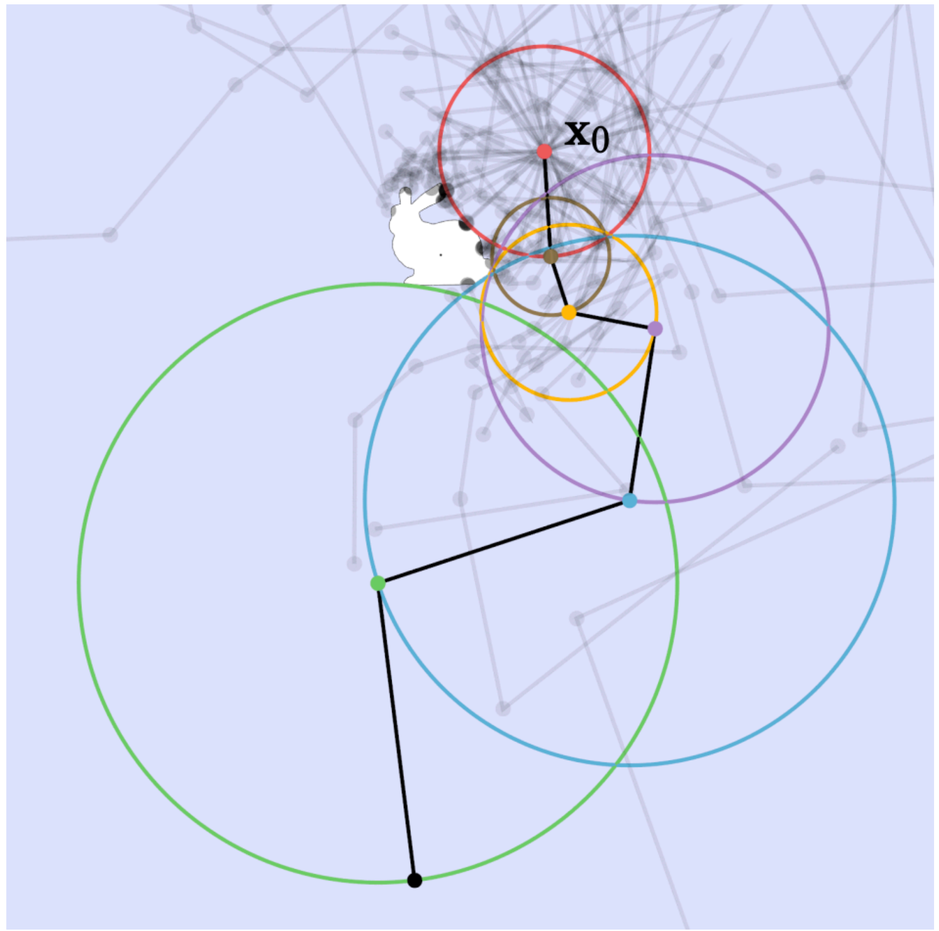
exterior problem



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

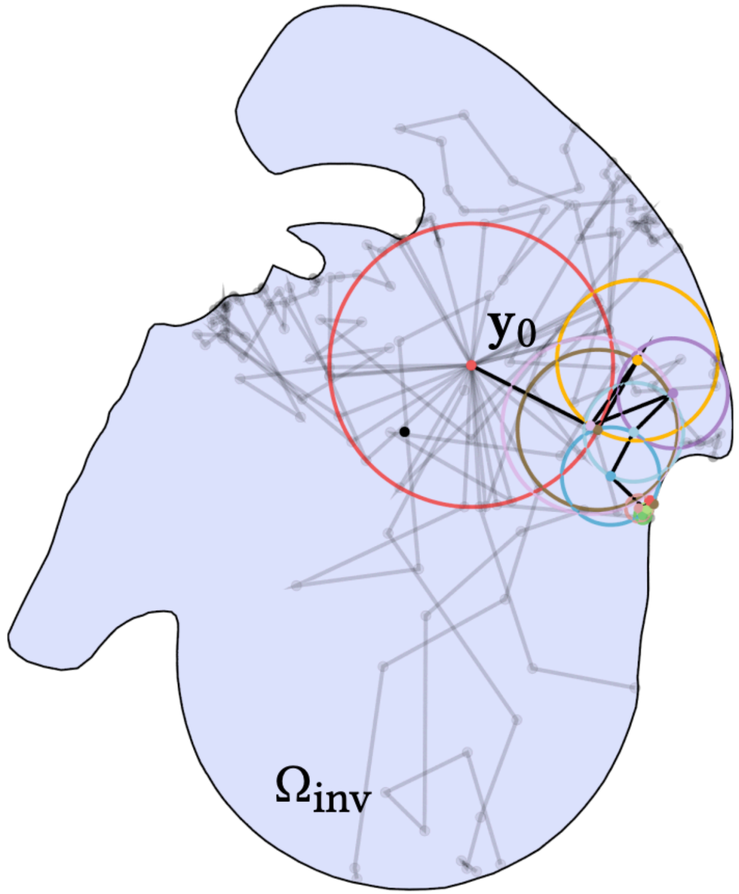
Kelvin transform for exterior problems

exterior problem



Invert via
Kelvin transform

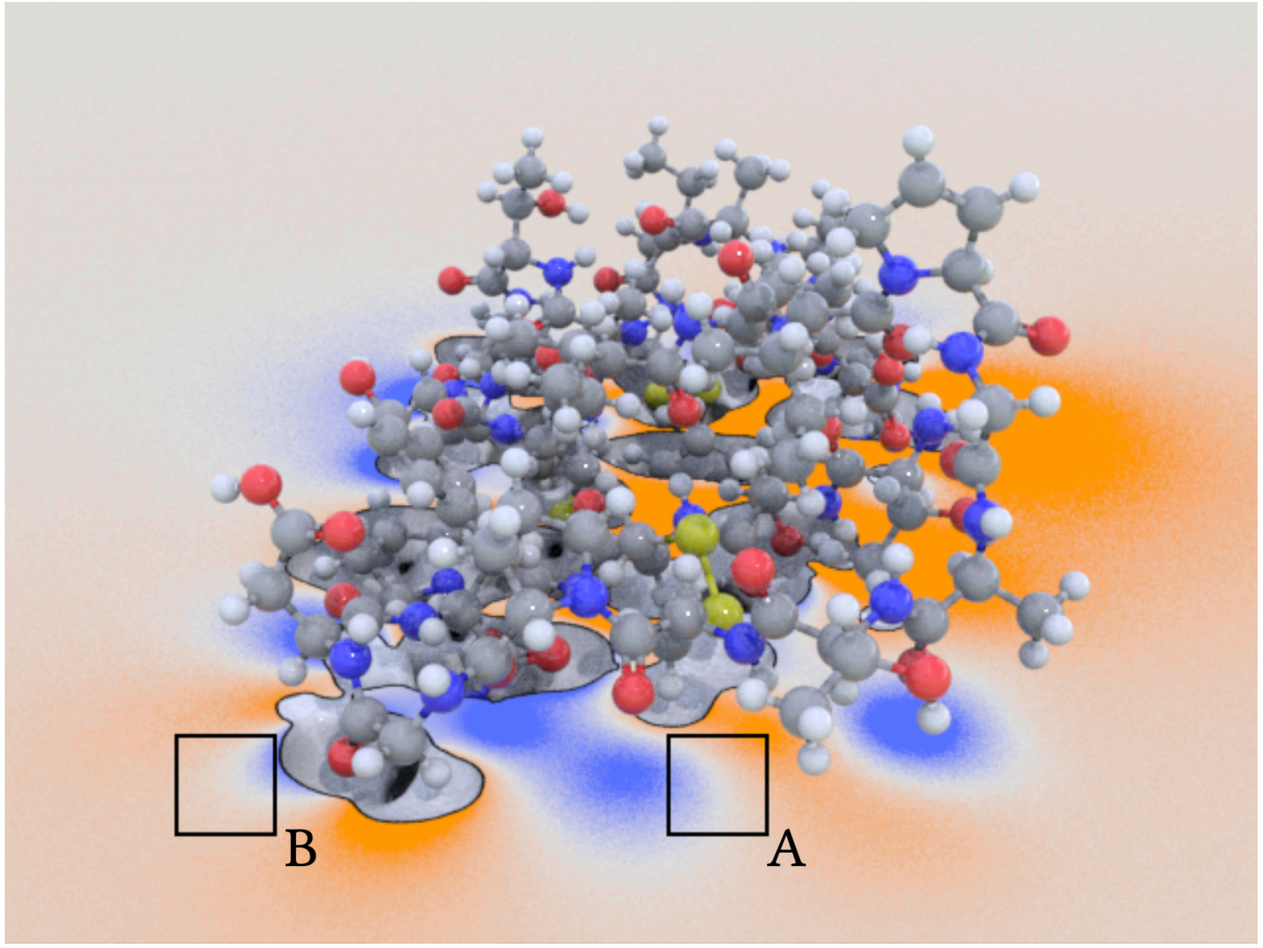
interior problem



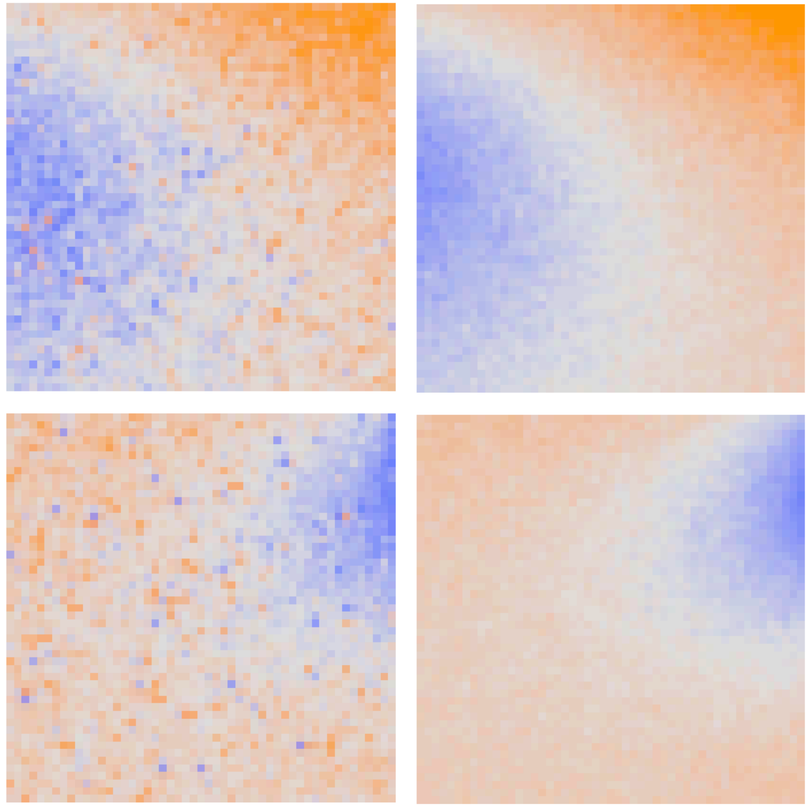
Nabizadeh et al., *"Kelvin Transformations for Simulations on Infinite Domains,"* SIGGRAPH 2021

Kelvin transform for exterior problems

Application: force evaluation for molecular dynamics simulation



(electrostatic potential on 1CRN protein)

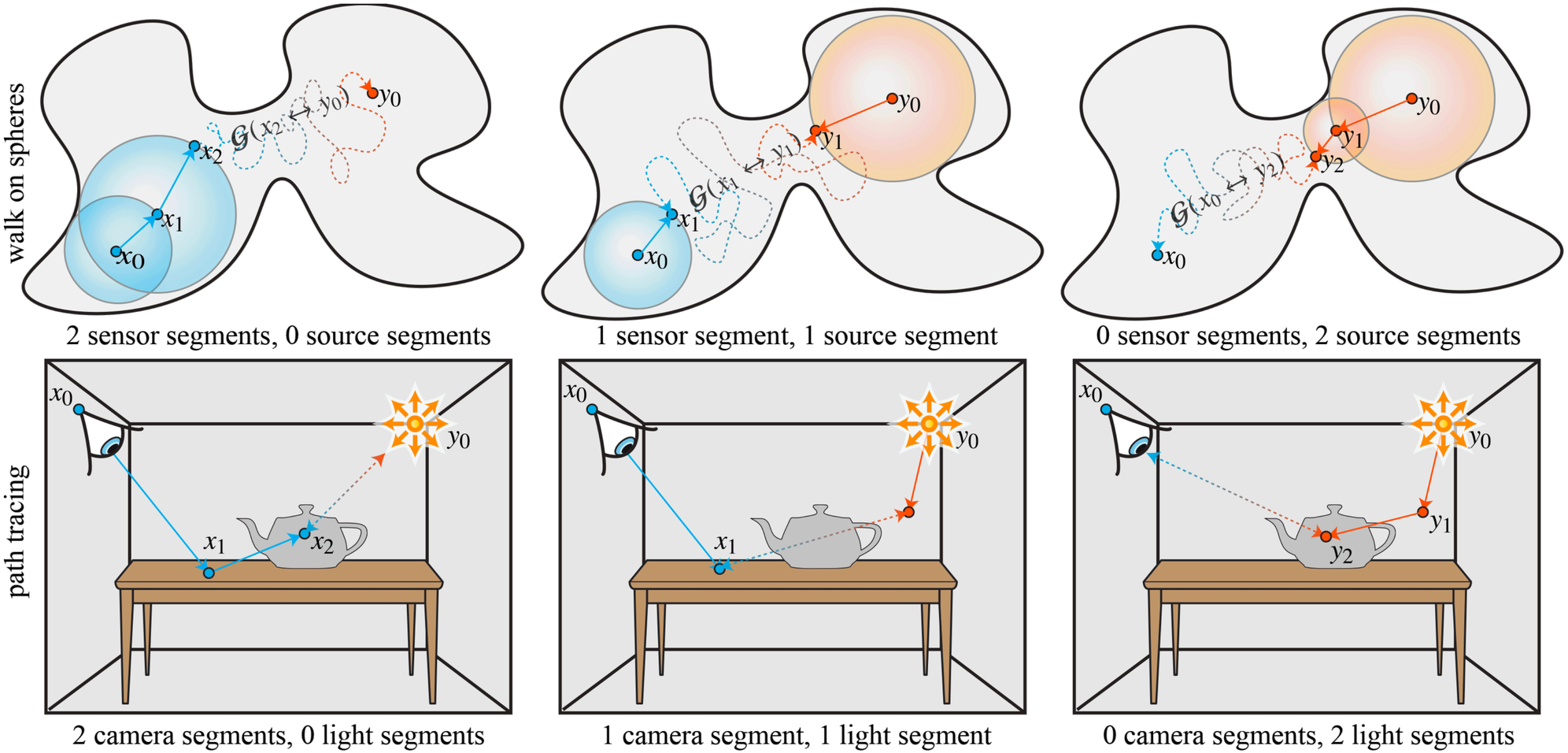


RUSSIAN ROULETTE

KELVIN TRANSFORM

Bidirectional walk on spheres

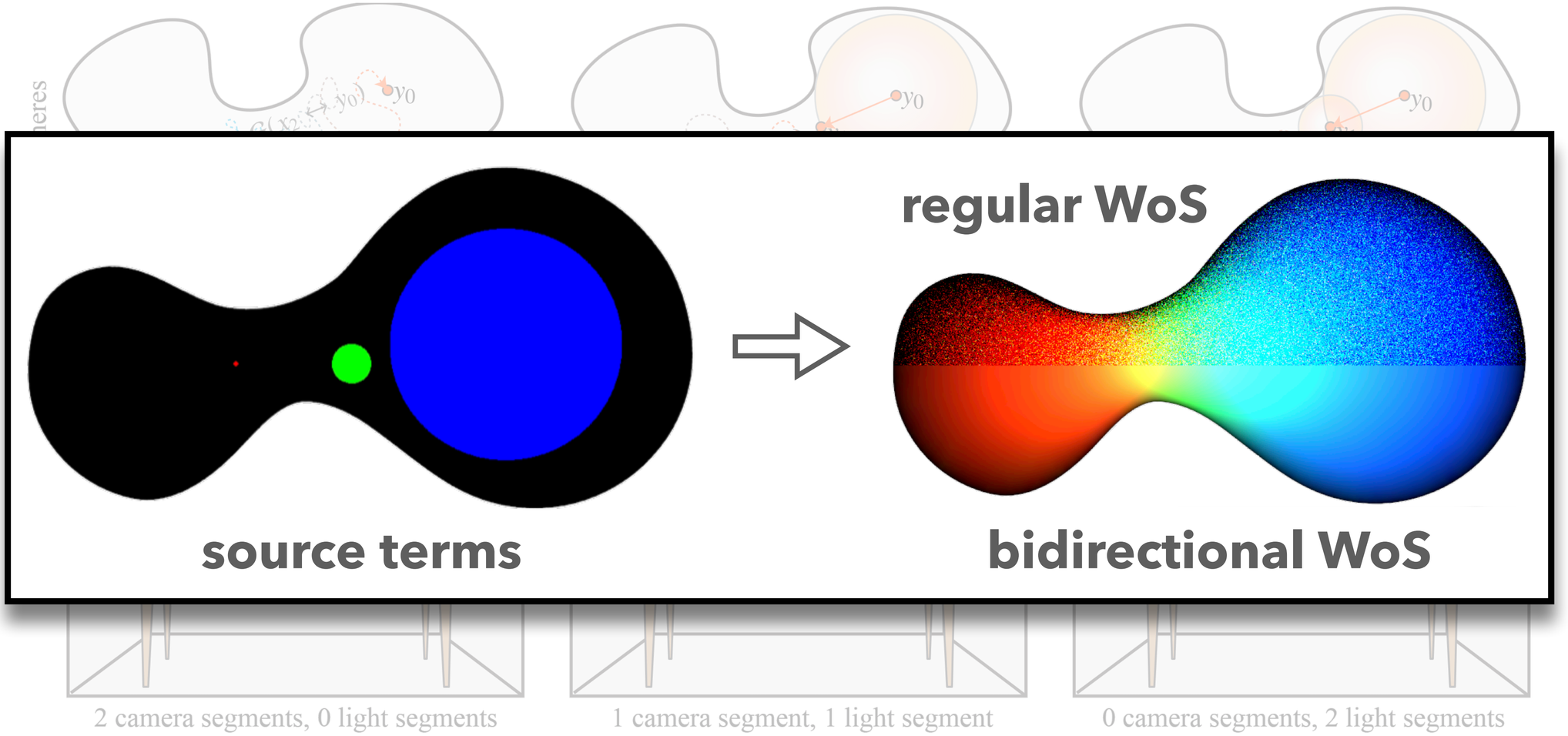
Key idea: start at source points, estimate *Green's function* at sensor points



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

Bidirectional walk on spheres

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

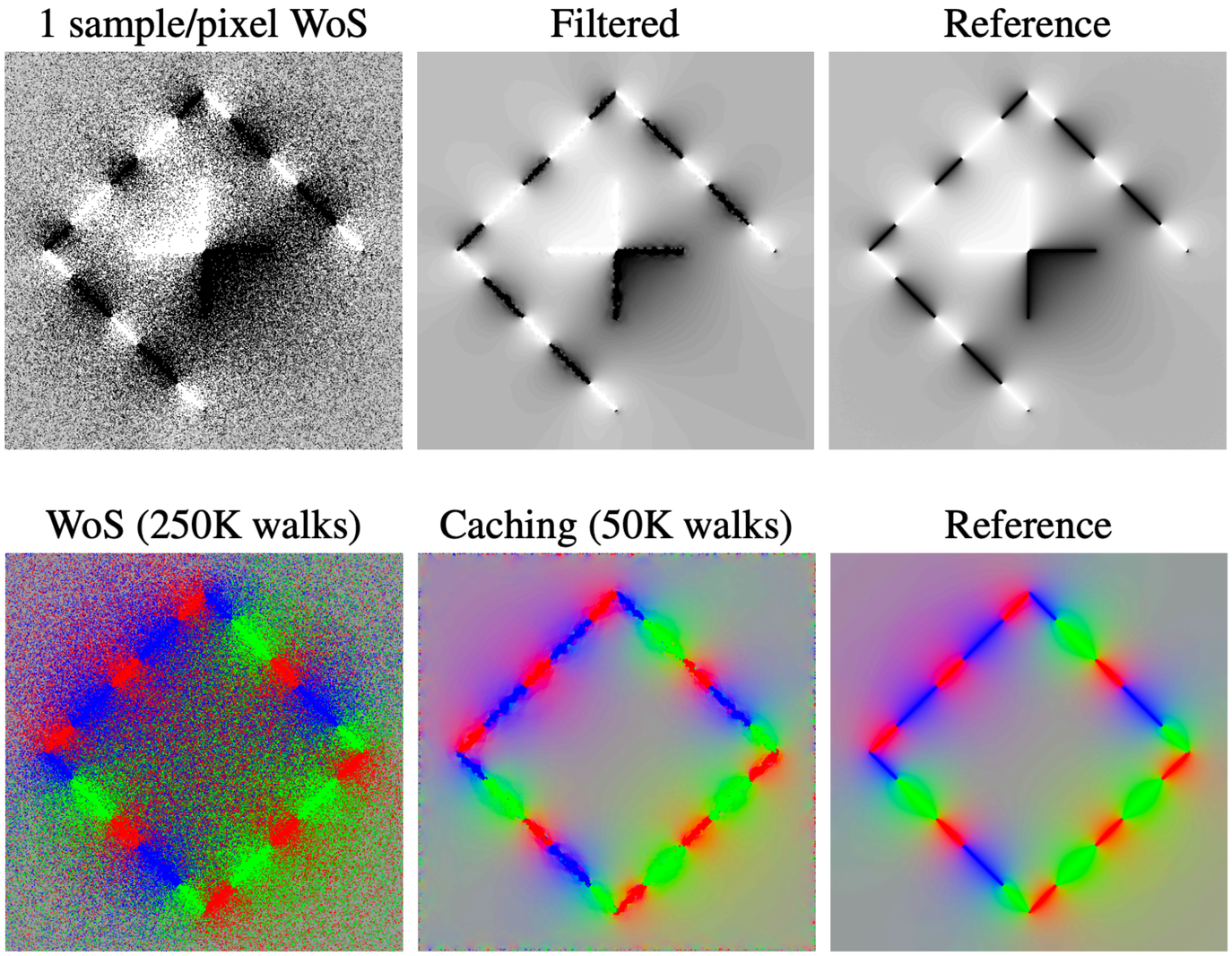


Qi, Seyb, Bitterli, Jarosz, *"A Bidirectional Formulation for Walk on Spheres,"* EGSR 2022

Sample reuse via Mean Value Caching

volumetric mean value property

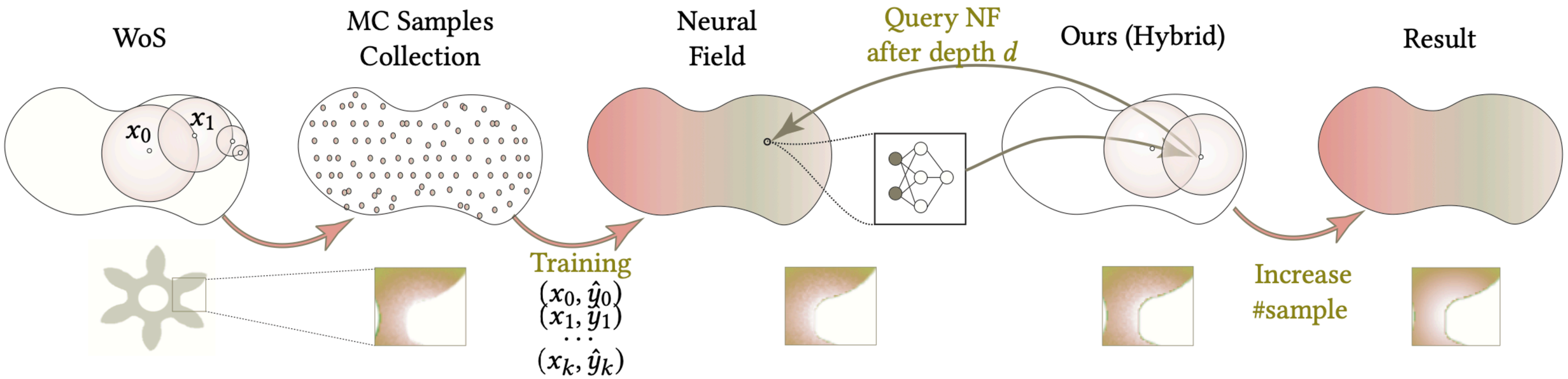
$$u(x) = \frac{1}{|B(x)|} \int_{B(x)} u(y)\ dy$$



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

Neural Caches

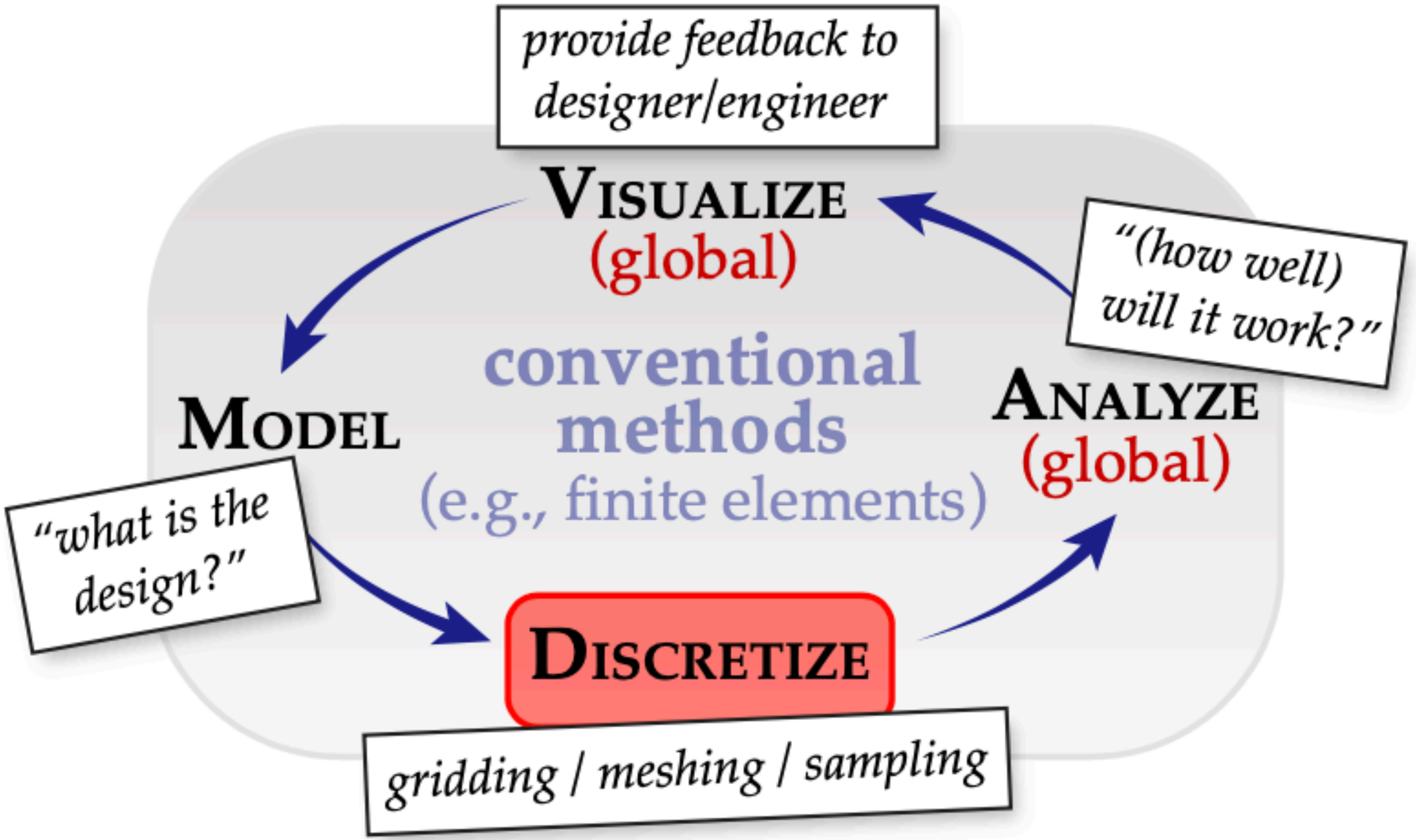
Key idea: Terminate walks early by training network to predict solution



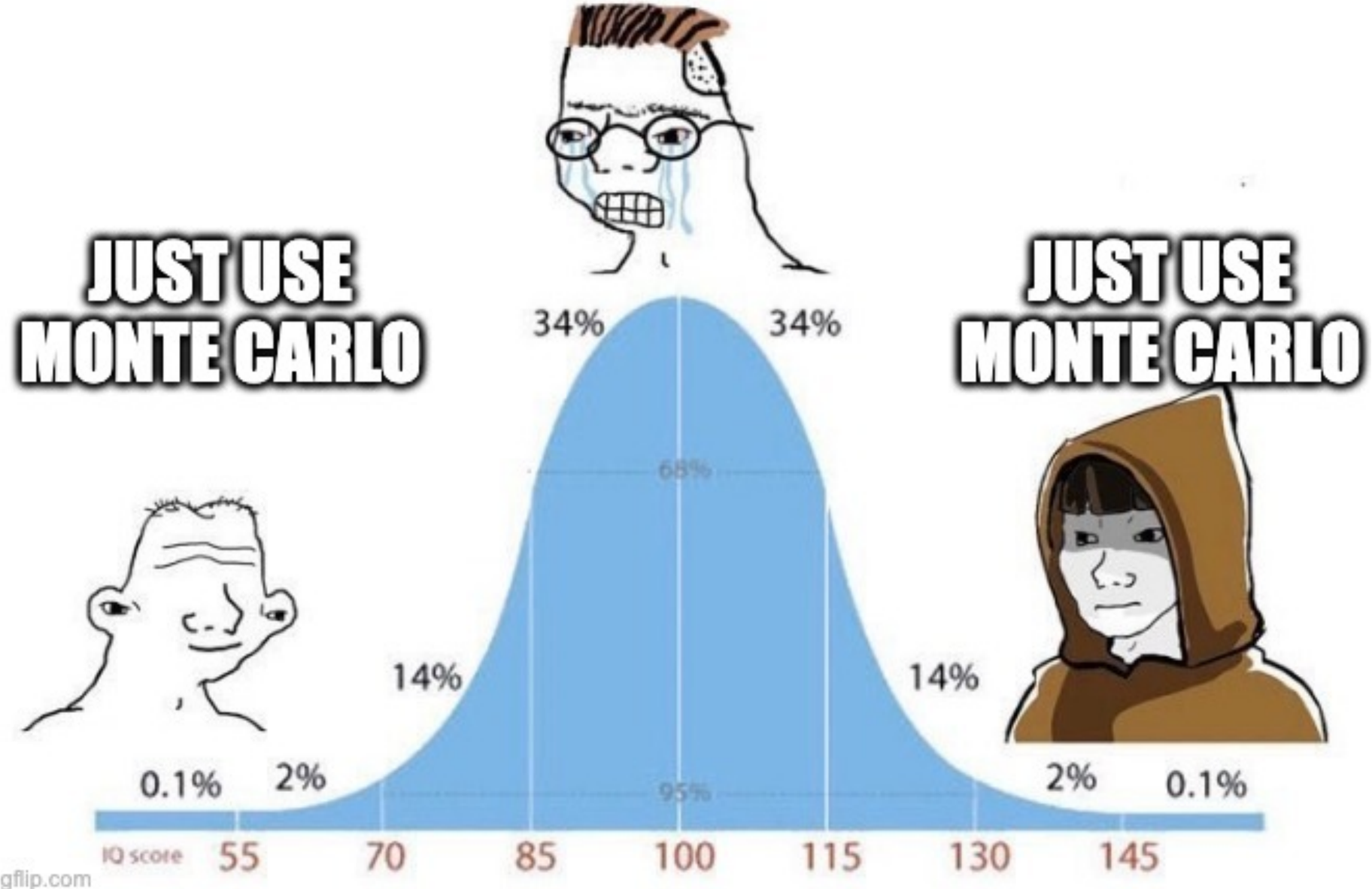
Li, Yang, Deng, De Sa, Hariharan, Marschner,
"Neural Caches for Monte Carlo Partial Differential Equation Solver," SIGGRAPH Asia 2023

Evaluation, Recent & Future work

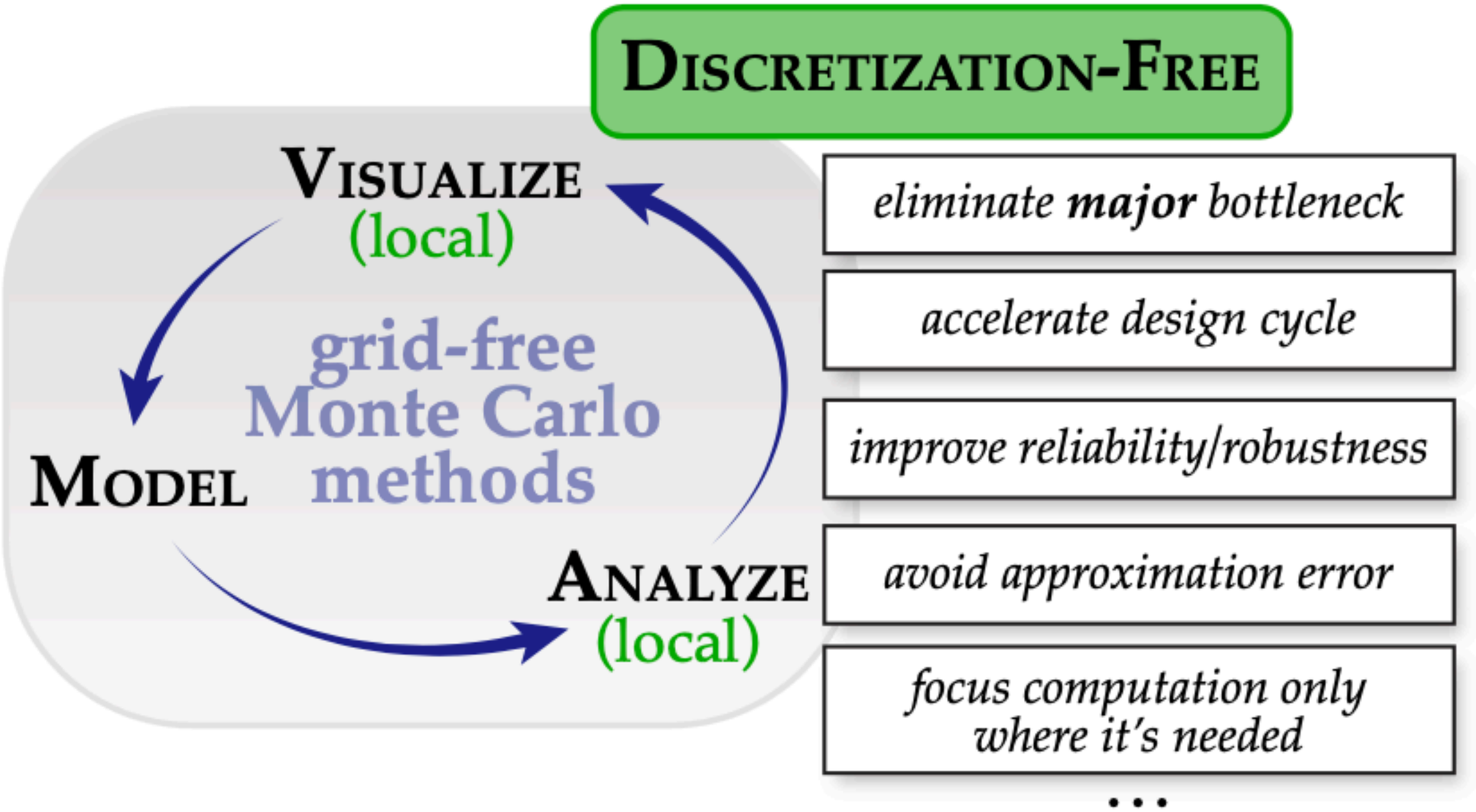
Global analysis – want to know solution everywhere?



MONTE CARLO IS TOO SLOW!



Local analysis – want to know solution in region of interest?



Comparison with conventional solvers

Comparison with conventional solvers

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…

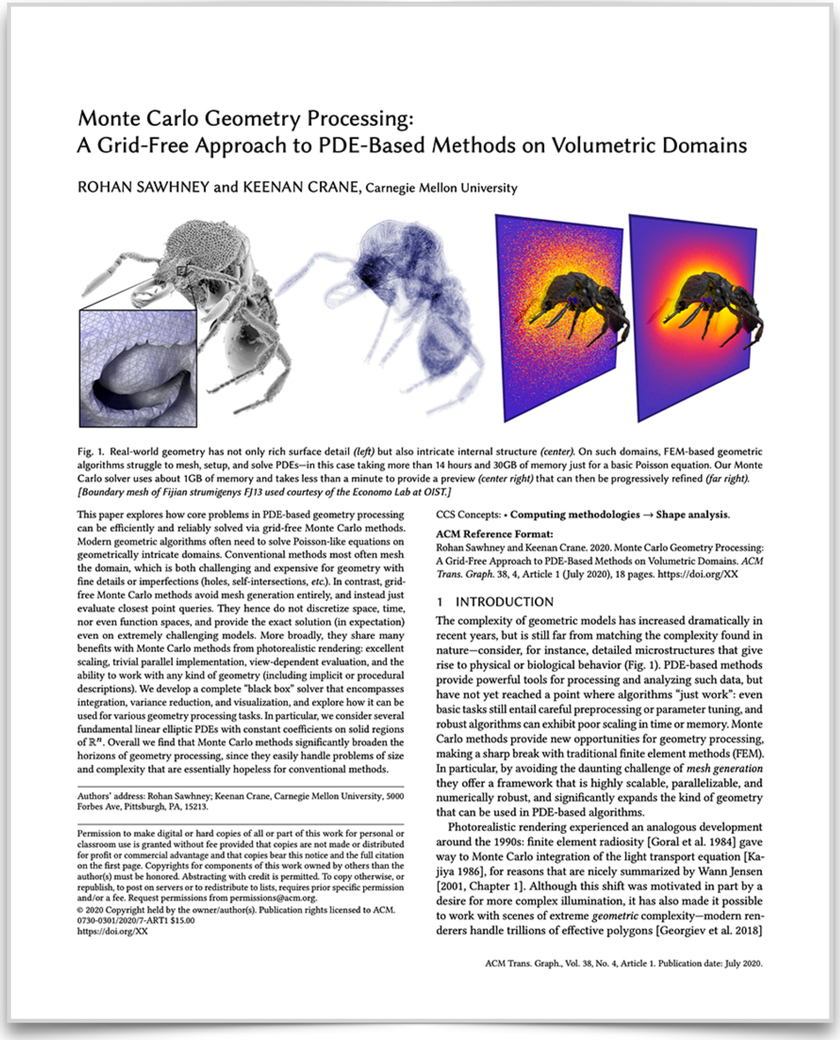
From: shirley@facility.cs.utah.edu
Subject: Re: What's wrong w/ Monte-Carlo methods?
To: cn1@irz301.inf.tu-dresden.de (Nguyen D.C.)
Date: Mon, 28 Oct 1996 08:31:22 -0700 (MST)
Cc: globillum@imag.fr, shirley@facility.cs.utah.edu (Peter Shirley)

…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



Monte Carlo Geometry Processing:
A Grid-Free Approach to PDE-Based Methods on Volumetric Domains

ROHAN SAWHNEY and KEENAN CRANE, Carnegie Mellon University

Fig. 1. Real-world geometry has not only rich surface detail *(left)* but also intricate internal structure *(center)*. On such domains, FEM-based geometric algorithms struggle to mesh, setup, and solve PDEs—in this case taking more than 14 hours and 30GB of memory just for a basic Poisson equation. Our Monte Carlo solver uses about 1GB of memory and takes less than a minute to provide a preview *(center right)* that can then be progressively refined *(far right)*. *[Boundary mesh of Fijian strumigenys FJ13 used courtesy of the Economo Lab at OIST.]*

This paper explores how core problems in PDE-based geometry processing can be efficiently and reliably solved via grid-free Monte Carlo methods. Modern geometric algorithms often need to solve Poisson-like equations on geometrically intricate domains. Conventional methods most often mesh the domain, which is both challenging and expensive for geometry with fine details or imperfections (holes, self-intersections, *etc.*). In contrast, grid-free Monte Carlo methods avoid mesh generation entirely, and instead just evaluate closest point queries. They hence do not discretize space, time, nor even function spaces, and provide the exact solution (in expectation) even on extremely challenging models. More broadly, they share many benefits with Monte Carlo methods from photorealistic rendering: excellent scaling, trivial parallel implementation, view-dependent evaluation, and the ability to work with any kind of geometry (including implicit or procedural descriptions). We develop a complete "black box" solver that encompasses integration, variance reduction, and visualization, and explore how it can be used for various geometry processing tasks. In particular, we consider several fundamental linear elliptic PDEs with constant coefficients on solid regions of $\mathbb{R}^n$. Overall we find that Monte Carlo methods significantly broaden the horizons of geometry processing, since they easily handle problems of size and complexity that are essentially hopeless for conventional methods.

Authors' address: Rohan Sawhney; Keenan Crane, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
0730-0301/2020/7-ART1 $15.00
https://doi.org/XX

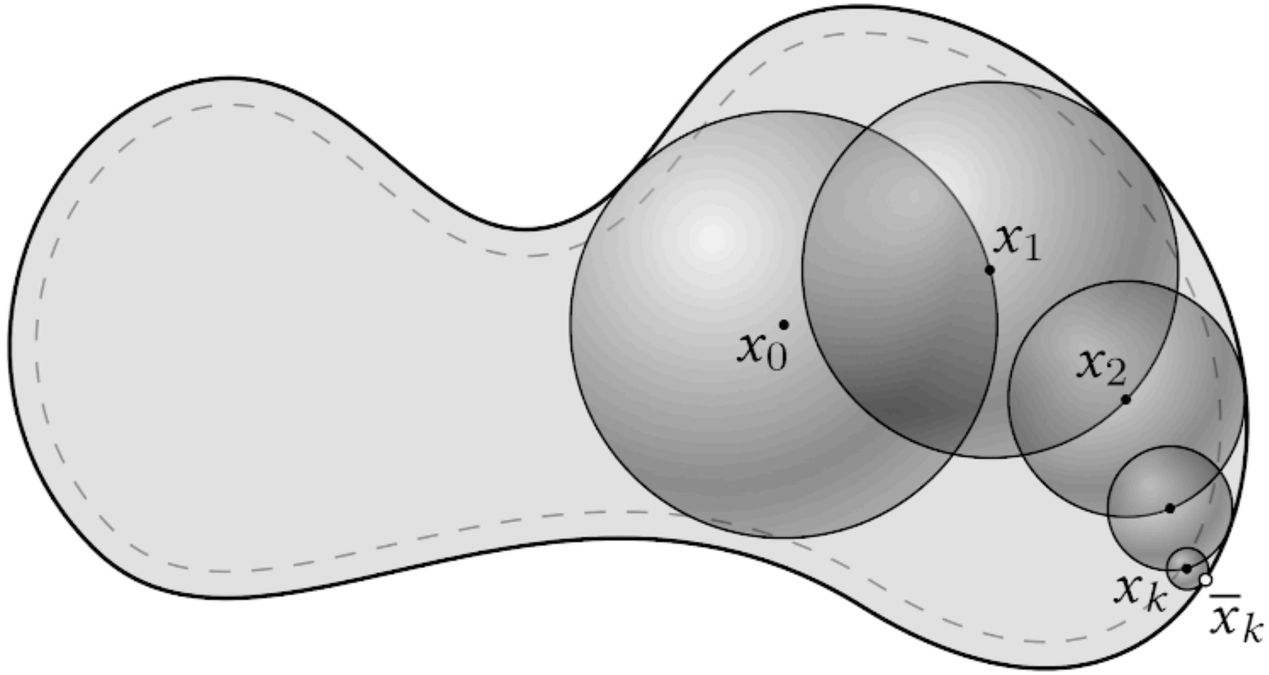
CCS Concepts: • **Computing methodologies** → **Shape analysis**.

ACM Reference Format:
Rohan Sawhney and Keenan Crane. 2020. Monte Carlo Geometry Processing: A Grid-Free Approach to PDE-Based Methods on Volumetric Domains. *ACM Trans. Graph.* 38, 4, Article 1 (July 2020), 18 pages. https://doi.org/XX

1 INTRODUCTION

The complexity of geometric models has increased dramatically in recent years, but is still far from matching the complexity found in nature—consider, for instance, detailed microstructures that give rise to physical or biological behavior (Fig. 1). PDE-based methods provide powerful tools for processing and analyzing such data, but have not yet reached a point where algorithms "just work": even basic tasks still entail careful preprocessing or parameter tuning, and robust algorithms can exhibit poor scaling in time or memory. Monte Carlo methods provide new opportunities for geometry processing, making a sharp break with traditional finite element methods (FEM). In particular, by avoiding the daunting challenge of *mesh generation* they offer a framework that is highly scalable, parallelizable, and numerically robust, and significantly expands the kind of geometry that can be used in PDE-based algorithms.

Photorealistic rendering experienced an analogous development around the 1990s: finite element radiosity [Goral et al. 1984] gave way to Monte Carlo integration of the light transport equation [Kajiya 1986], for reasons that are nicely summarized by Wann Jensen [2001, Chapter 1]. Although this shift was motivated in part by a desire for more complex illumination, it has also made it possible to work with scenes of extreme *geometric* complexity—modern renderers handle trillions of effective polygons [Georgiev et al. 2018]

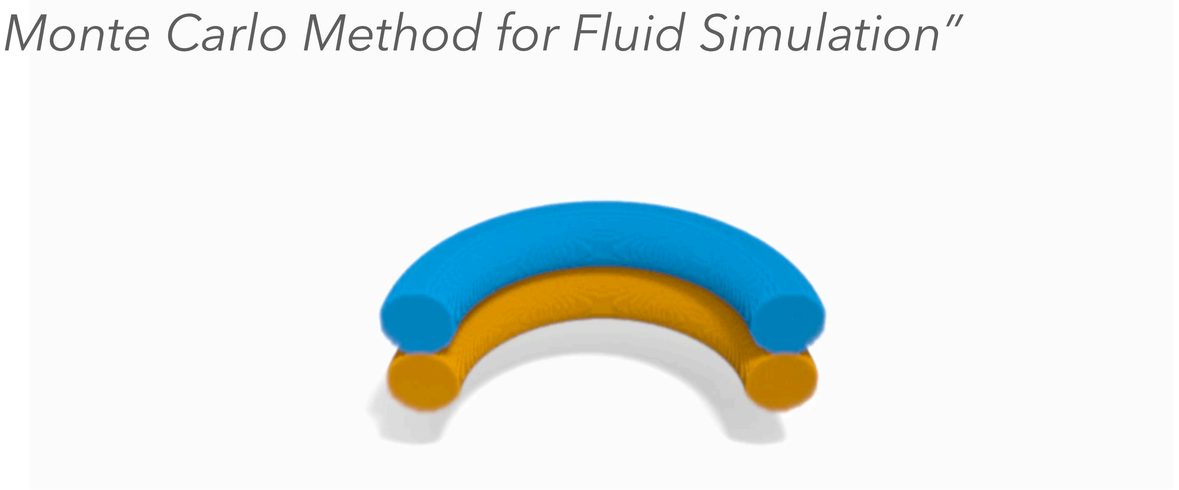


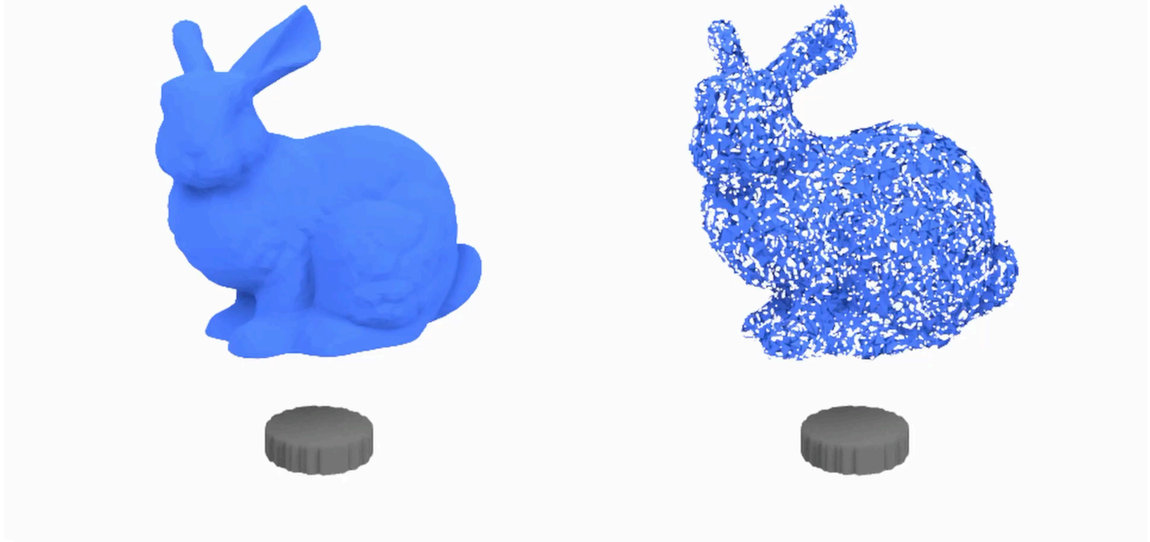
[Sawhney and Crane, SIGGRAPH 2020]

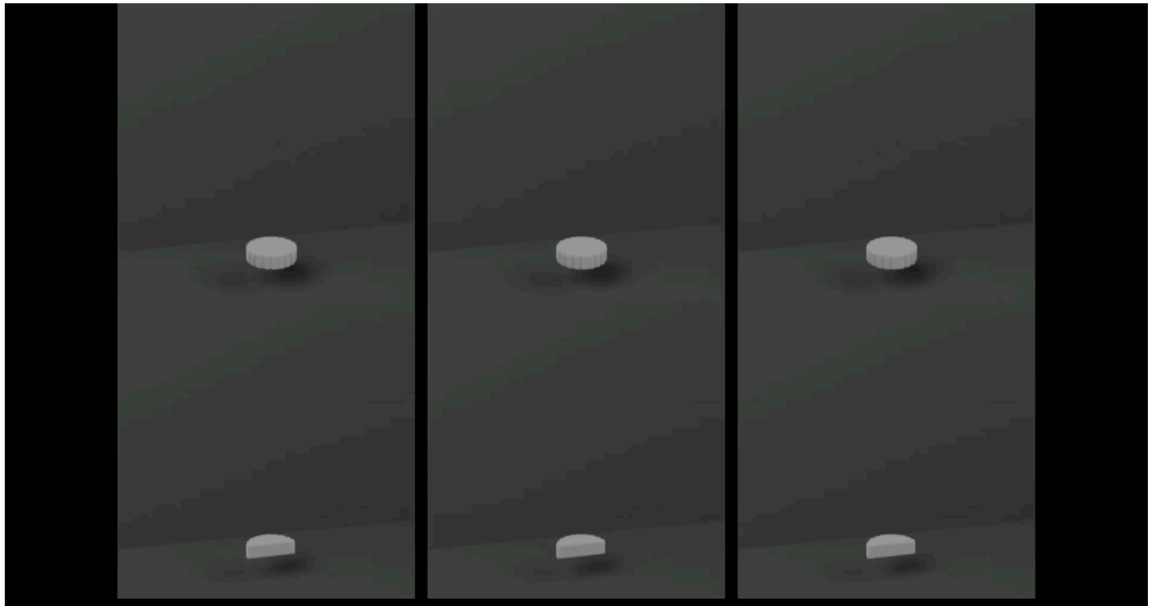
Monte Carlo fluid simulation

Rioux-Lavoie, Sugimoto, Özdemir, Shimada, Batty, Nowrouzezahari, Hachisuka (SIGGRAPH Asia 2022)

"A Monte Carlo Method for Fluid Simulation"



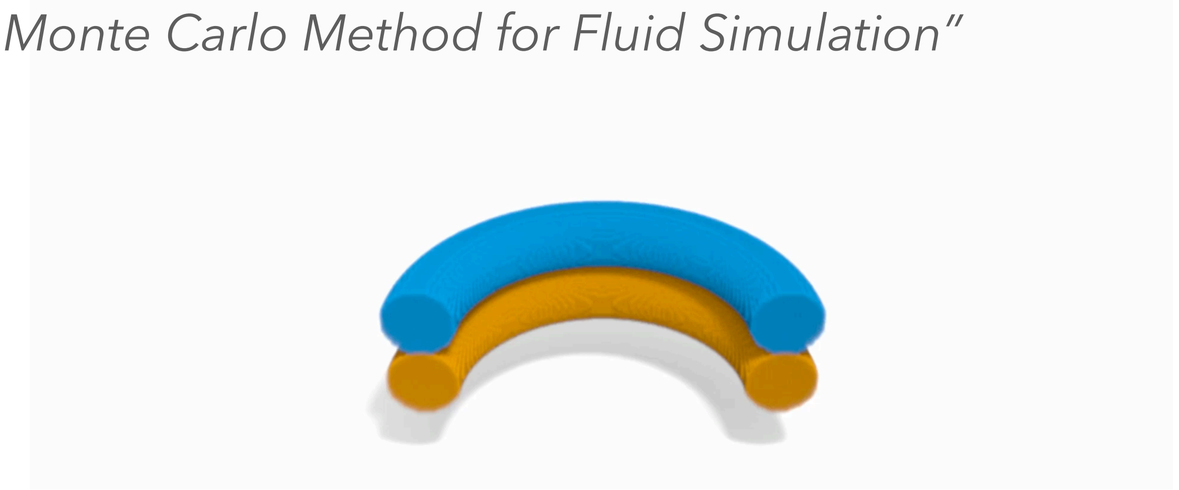


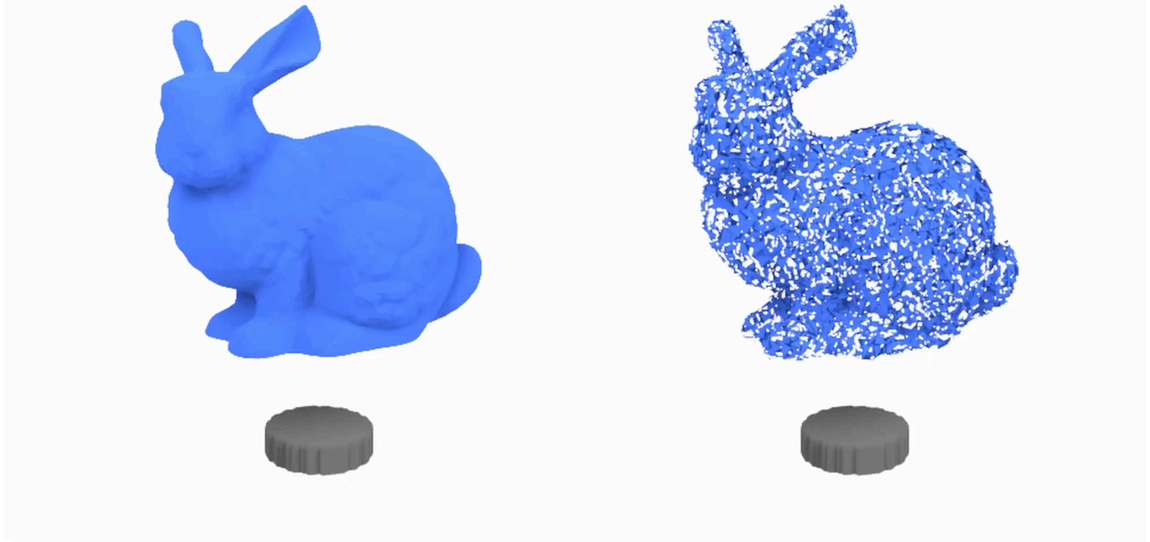


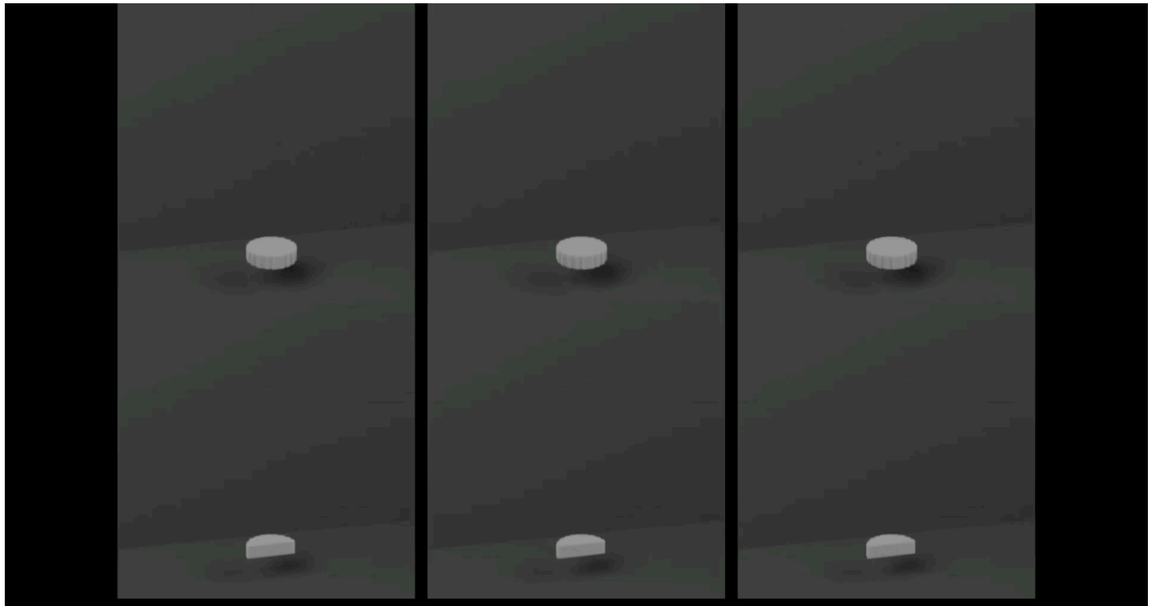
Monte Carlo fluid simulation

Rioux-Lavoie, Sugimoto, Özdemir, Shimada, Batty, Nowrouzezahari, Hachisuka (SIGGRAPH Asia 2022)

"A Monte Carlo Method for Fluid Simulation"

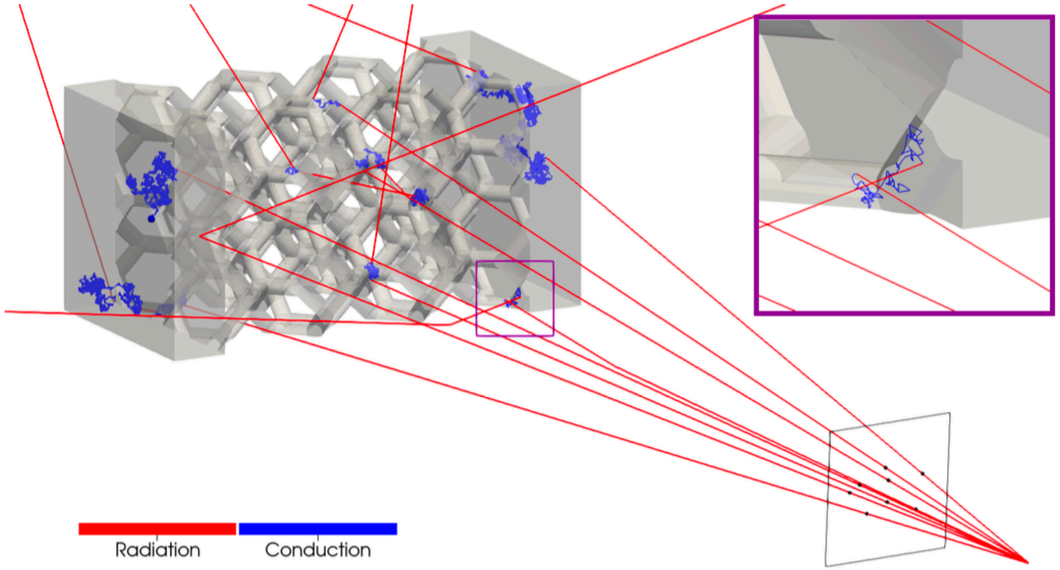


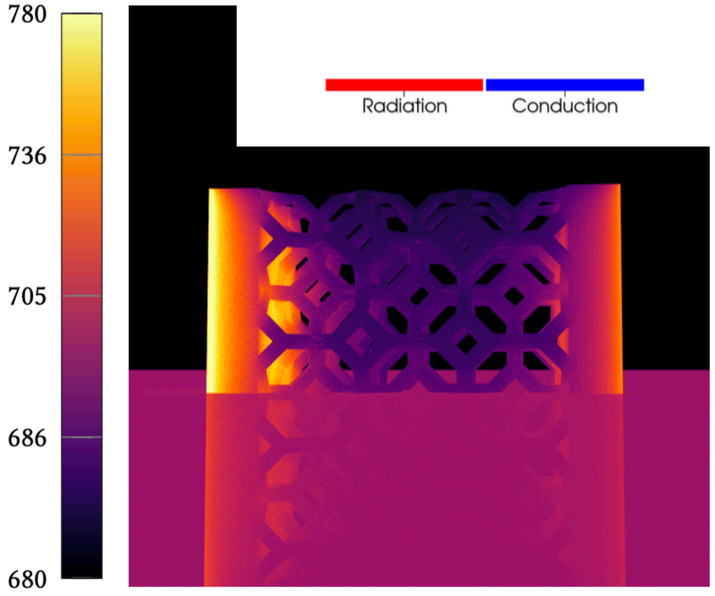


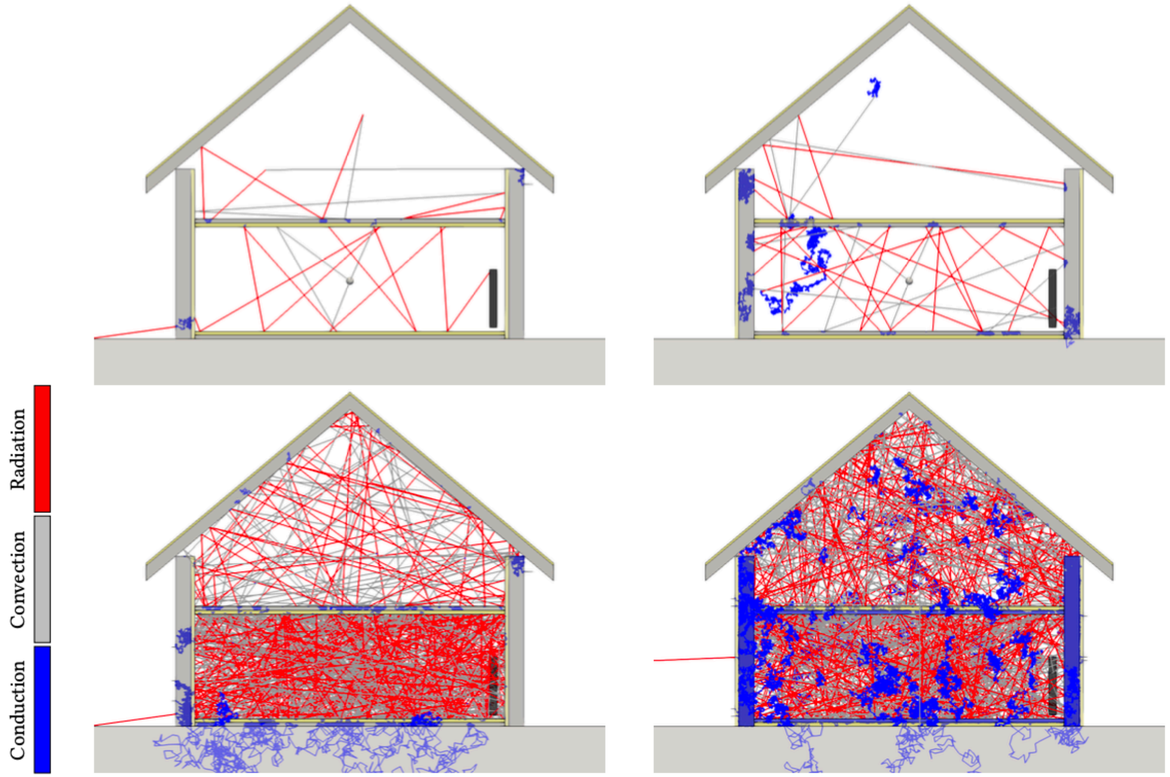


Monte Carlo Infrared Imaging

"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

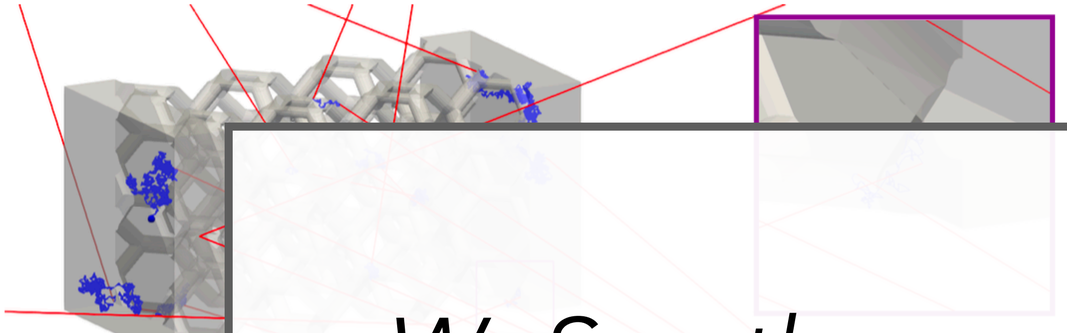






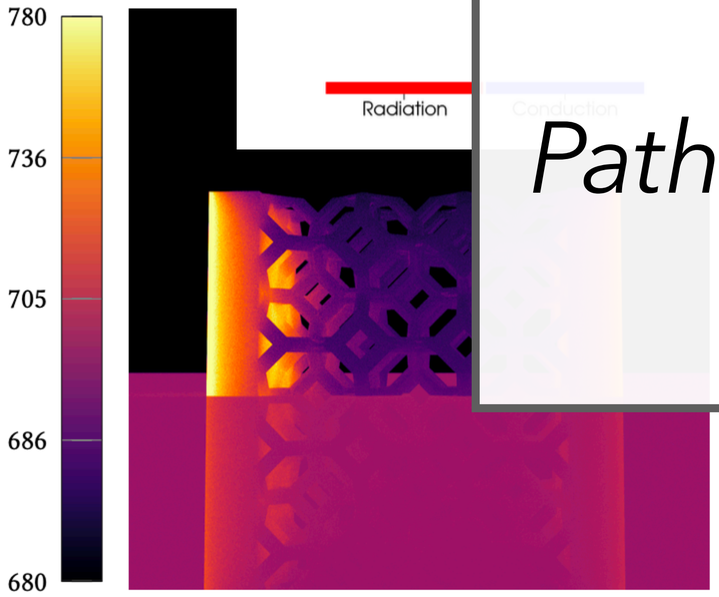
Monte Carlo Infrared Imaging

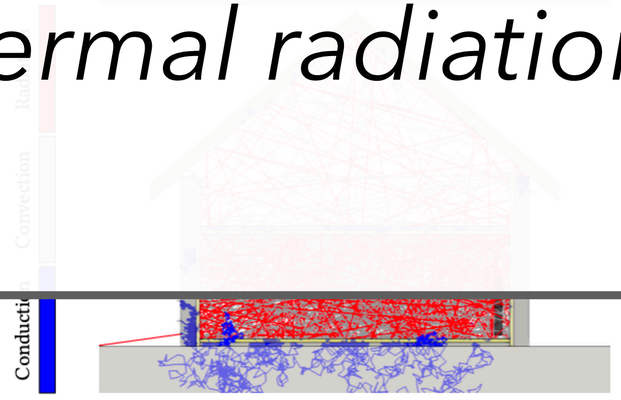
"Coupling Conduction, Convection and Radiative Transfer in a Single Path-Space: Application to Infrared Rendering"
Bati, Blanco, Coustet, Eymet, Forest, Fournier, Gautrais, Mellado, Paulin, Piaud, SIGGRAPH 2023

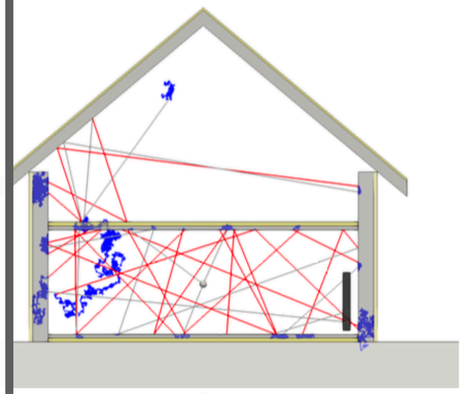


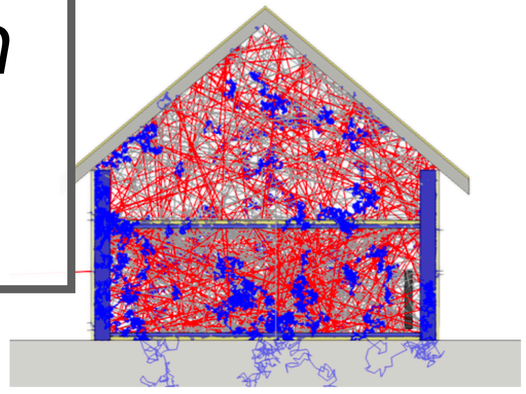
WoS ↔ thermal conduction

Path tracing ↔ thermal radiation









Creative uses

3D curve inflation

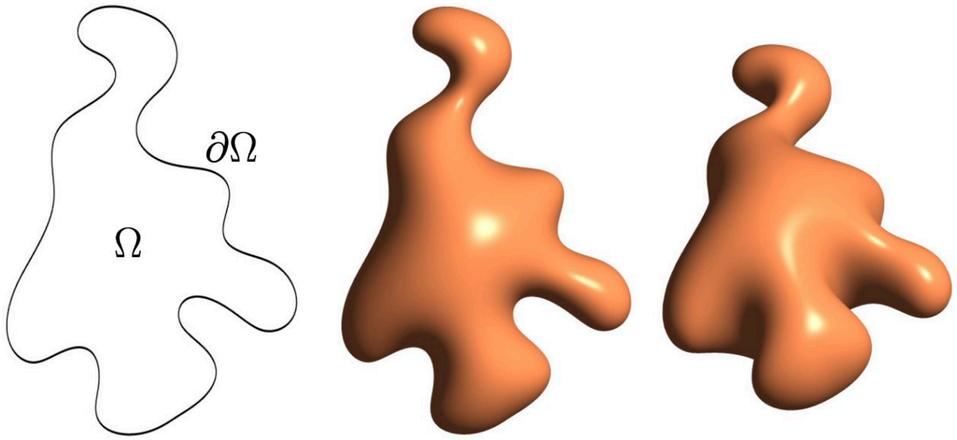


Figure 1: The curve (left) is inflated into a surface (middle). On the right is a rotated view of the result.

[Baran & Lehtinen 2009]

Credit: Alec Jacobson

Fractal generation

Credit: Mattias Malmer

Creative uses

3D curve inflation

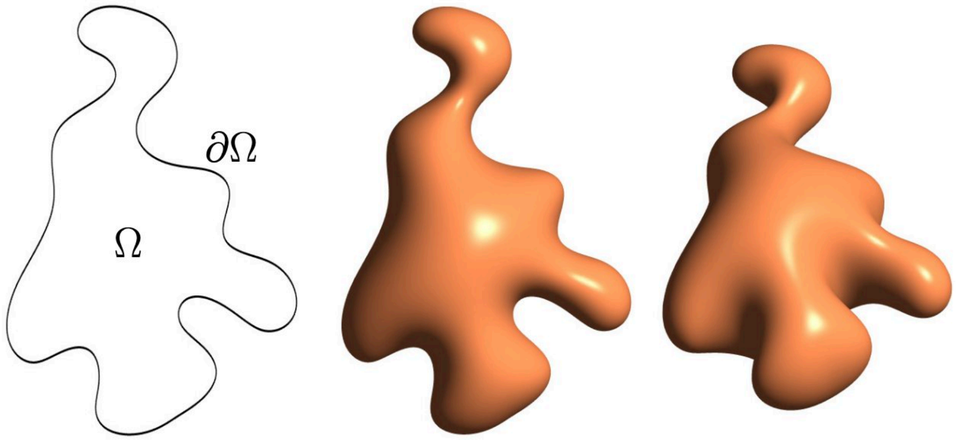


Figure 1: The curve (left) is inflated into a surface (middle). On the right is a rotated view of the result.

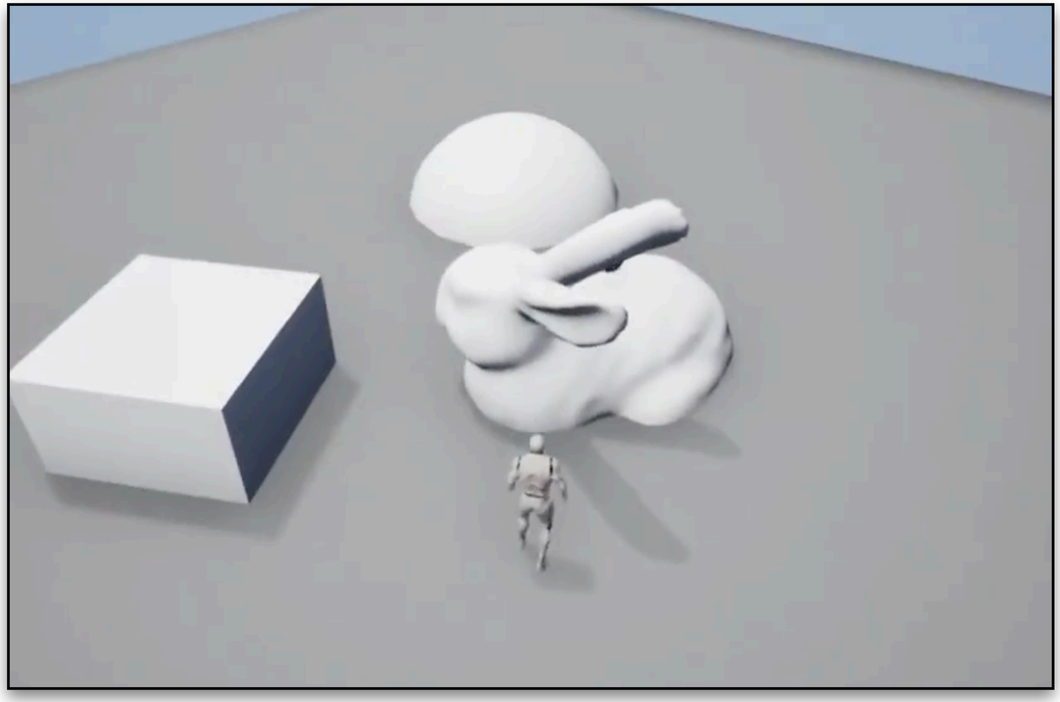
[Baran & Lehtinen 2009]

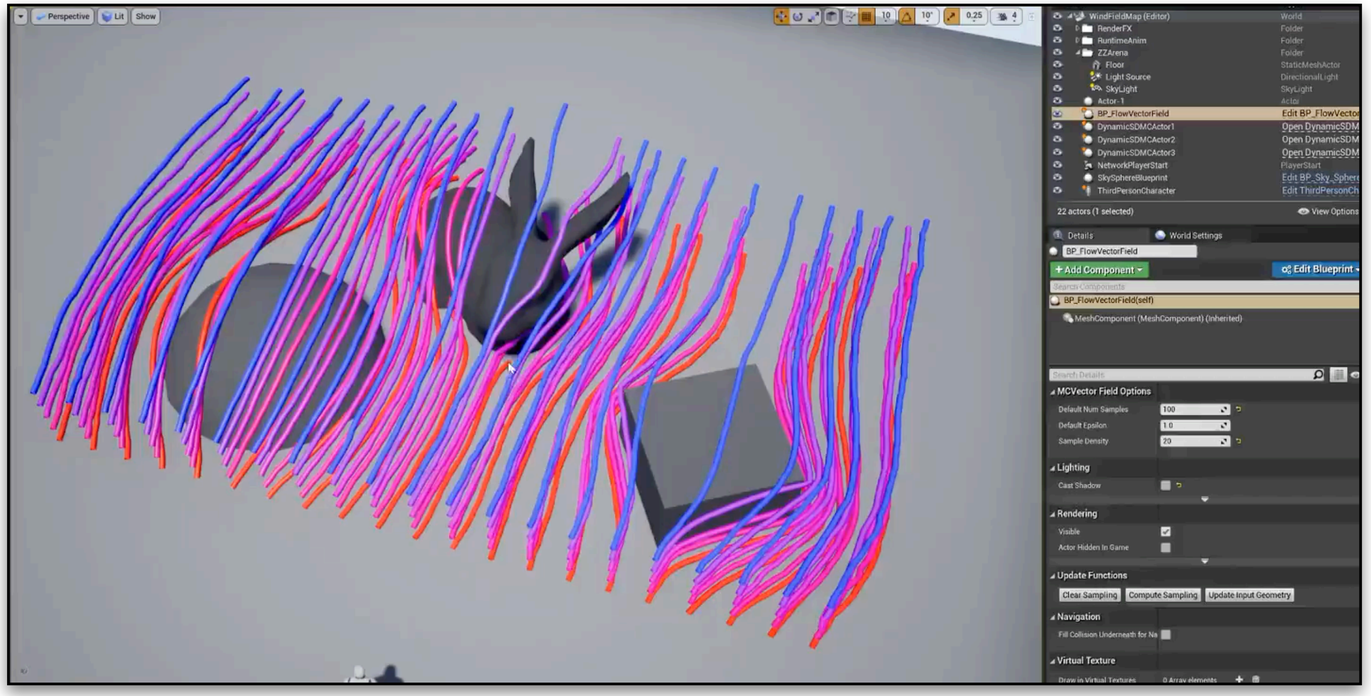
Credit: Alec Jacobson

Fractal generation

Credit: Mattias Malmer

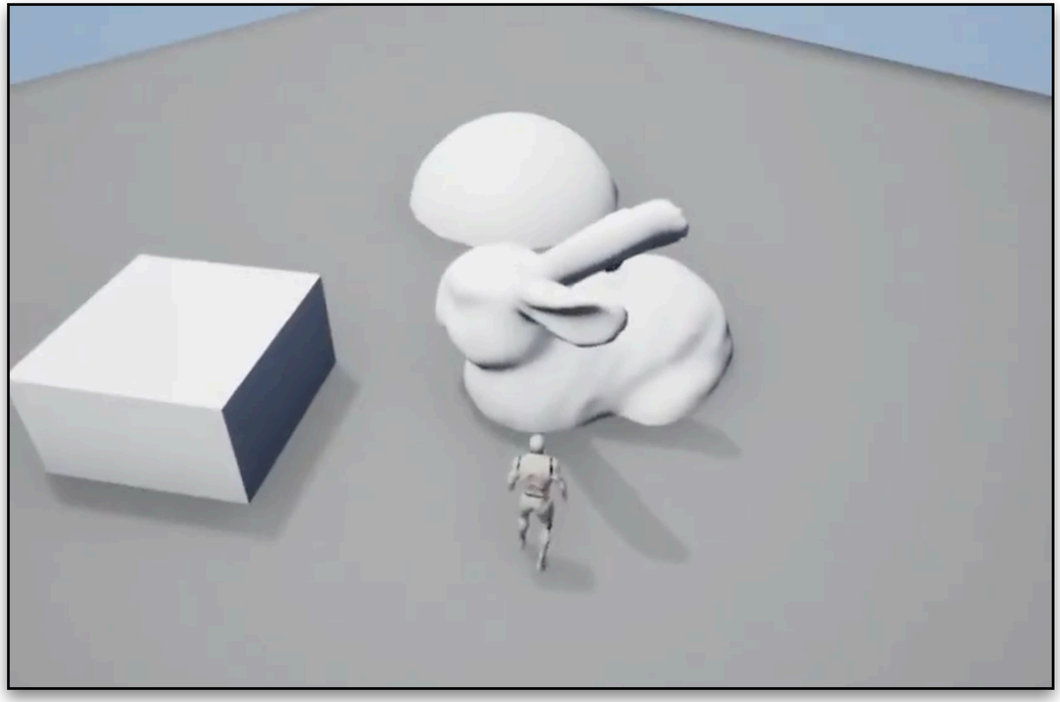
Agent path planning

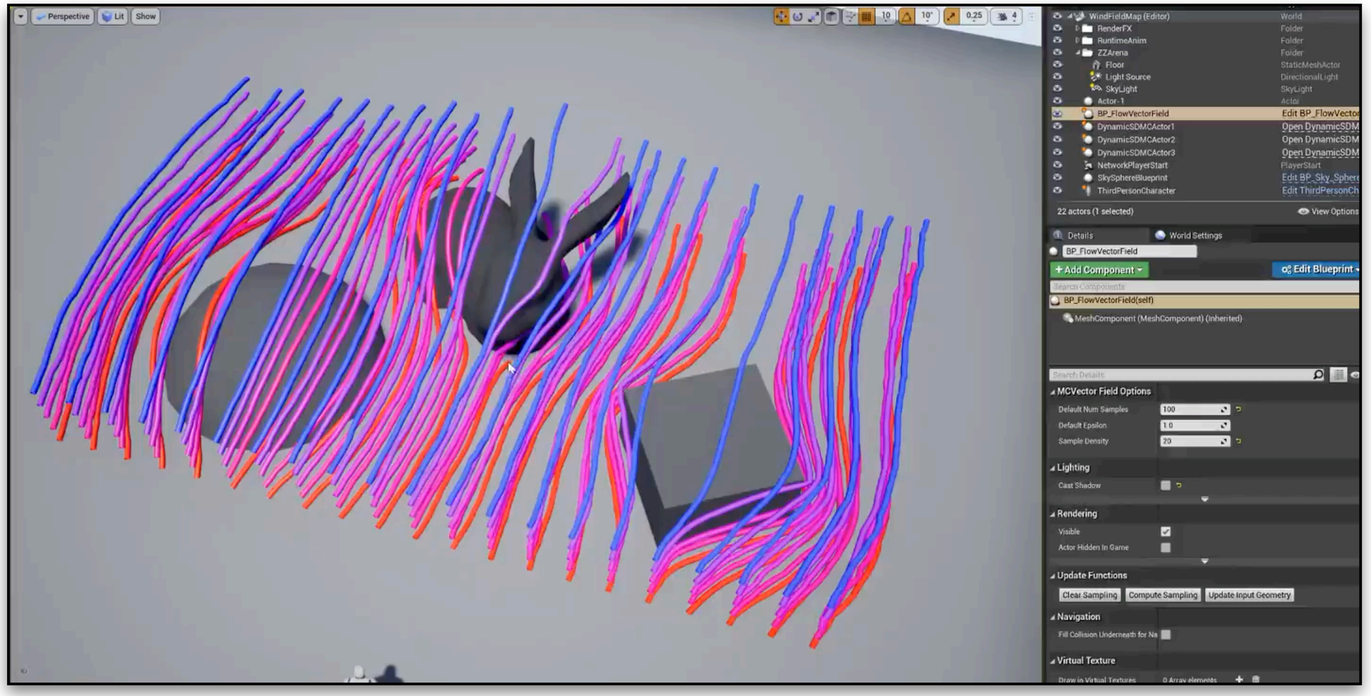




Credit: Ryan Schmidt

Agent path planning



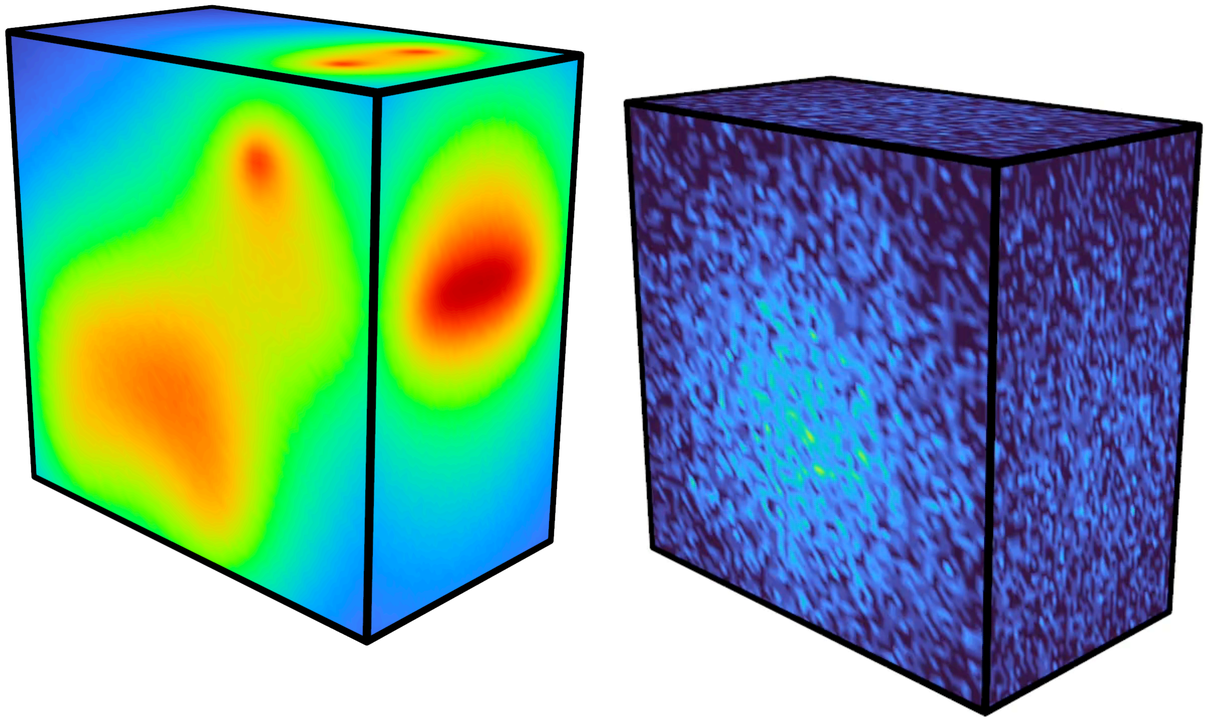


Credit: Ryan Schmidt

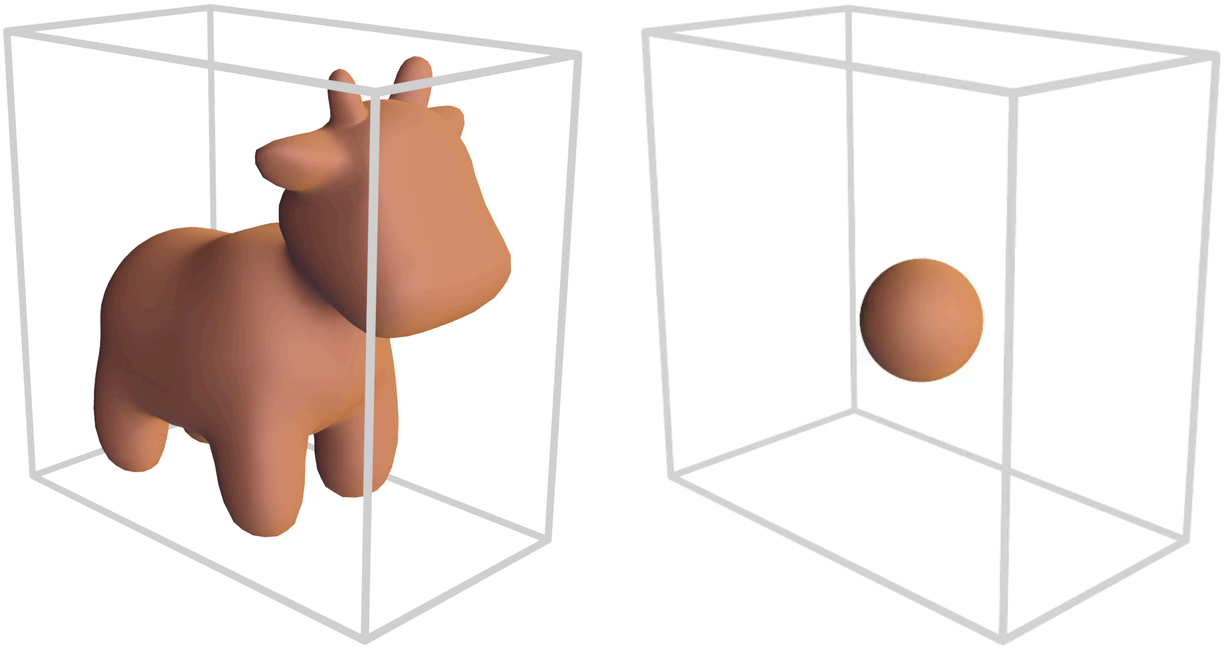
Differentiable solvers for PDE-based shape optimization

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

reference optimized reference optimized



solution



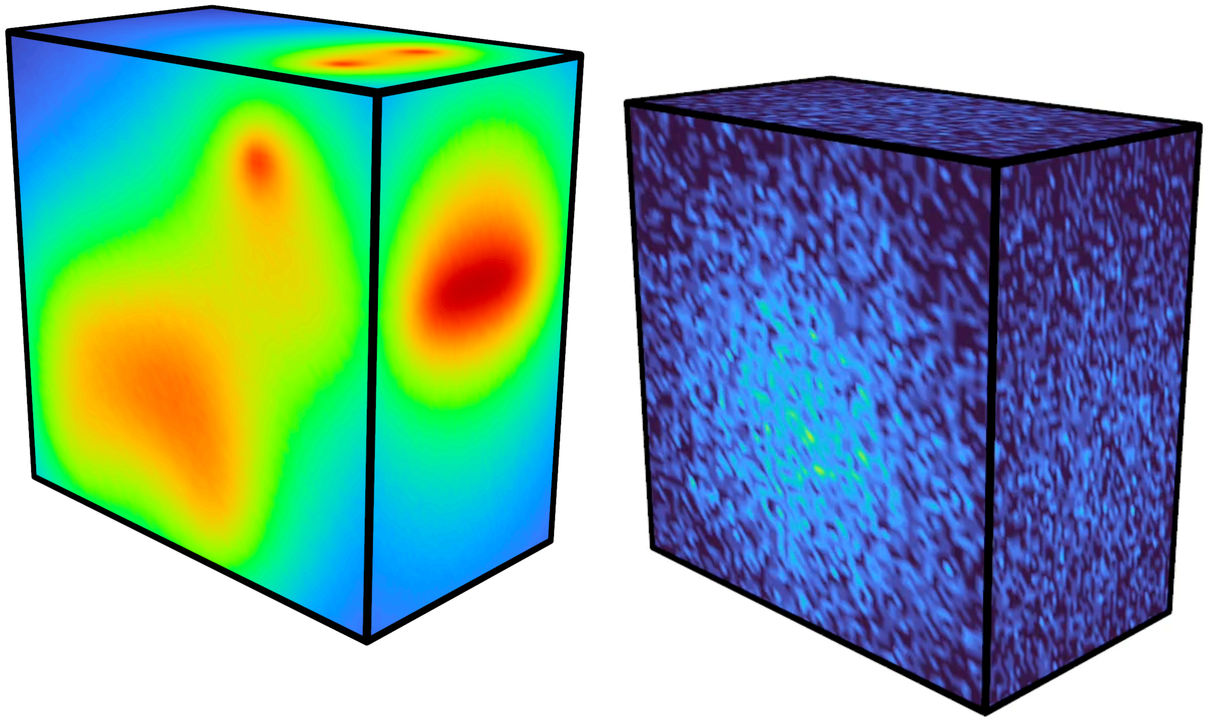
boundary

0 1

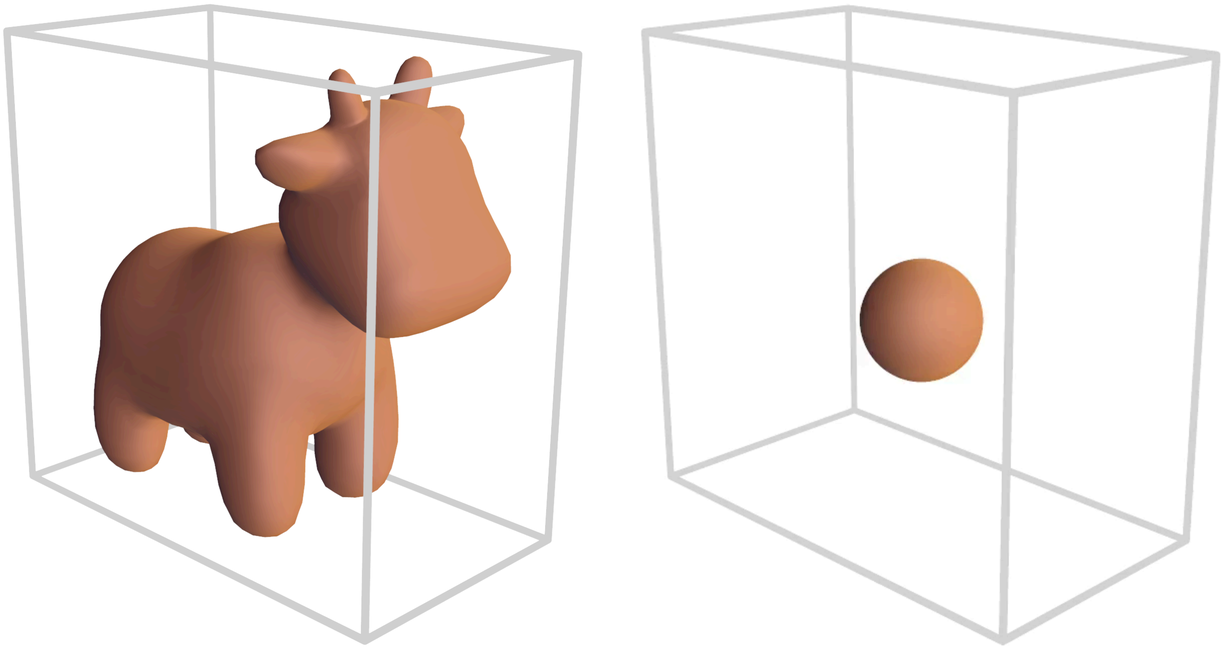
Differentiable solvers for PDE-based shape optimization

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

reference optimized reference optimized



solution



boundary

0 1

Next Steps

Extend WoS to more PDEs

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

Research → Production

– High-performance GPU implementation

– Variance reduction: denoising & supersampling

“Real-world” applications

– exploit unique capabilities of Monte Carlo

END.