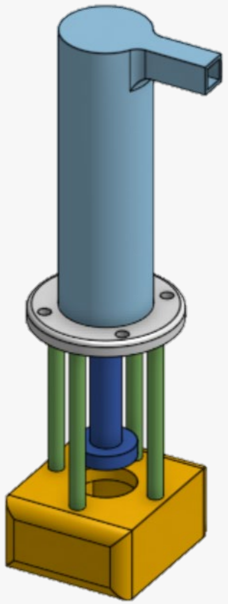
SIGGRAPH ASIA 2020 VIRTUAL
Pre-recorded sessions: From 4 December 2020
Live sessions: 10 – 13 December 2020
SA2020.SIGGRAPH.ORG
#SIGGRAPHAsia | #SIGGRAPHAsia2020

Functional Optimization of Fluidic Devices with Differentiable Stokes Flow

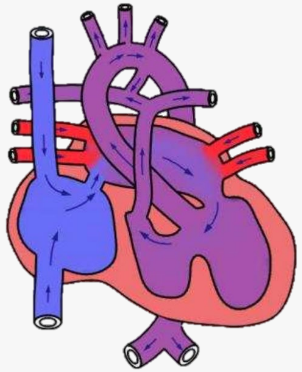
Tao Du	*MIT CSAIL*
Kui Wu	*MIT CSAIL*
Andrew Spielberg	*MIT CSAIL*
Wojciech Matusik	*MIT CSAIL*
Bo Zhu	*Dartmouth College*
Eftychios Sifakis	*University of Wisconsin-Madison*

Motivation

Fluidic devices are key components for a variety of products

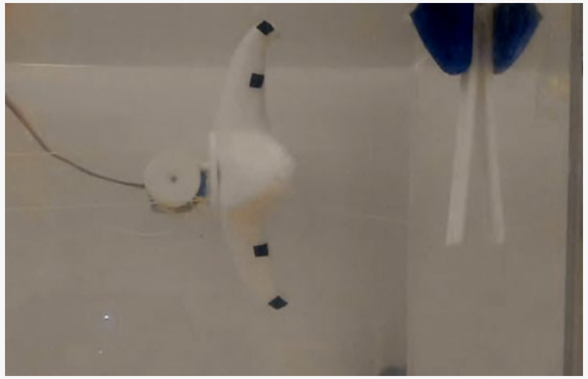


Developing
hydraulic actuators



Designing
medical devices

Fabricating underwater
soft robots



Motivation

SIGGRAPH ASIA 2020 VIRTUAL

However, designing fluidic devices is challenging

- The design space is large and non-trivial to parametrize

- The dynamics is computationally expensive due to the solid-fluid coupling

- Search for an optimal solution is challenging

Motivation

However, designing fluidic devices is challenging

- The design space is large and non-trivial to parametrize
- The dynamics is computationally expensive due to the solid-fluid coupling
- Search for an optimal solution is challenging

Motivation

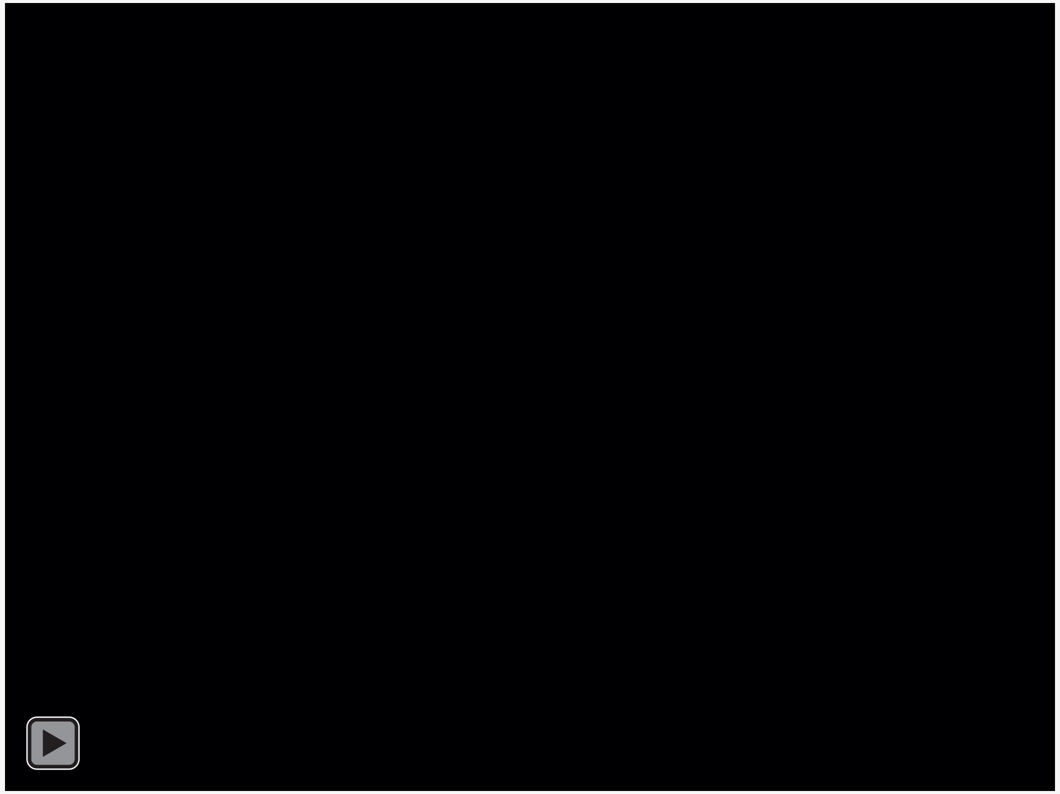
SIGGRAPH ASIA 2020 VIRTUAL

However, designing fluidic devices is challenging

- The design space is large and non-trivial to parametrize
- The dynamics is computationally expensive due to the solid-fluid coupling
- Search for an optimal solution is challenging

Motivation

We propose a computational design method for fluidic devices

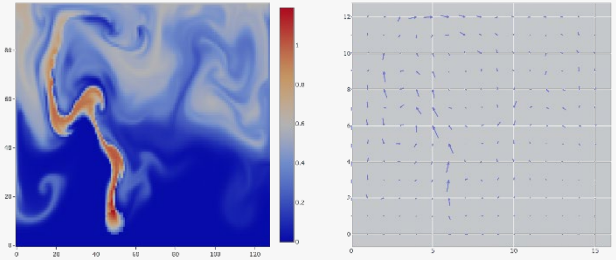


Related Work

SIGGRAPH ASIA 2020 VIRTUAL

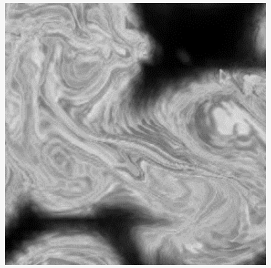
Fluid control

SIGGRAPH 04’

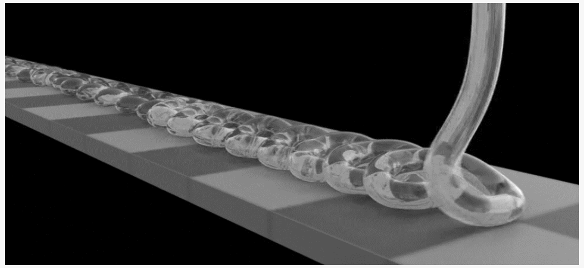


ICLR 20’

Fluid simulation



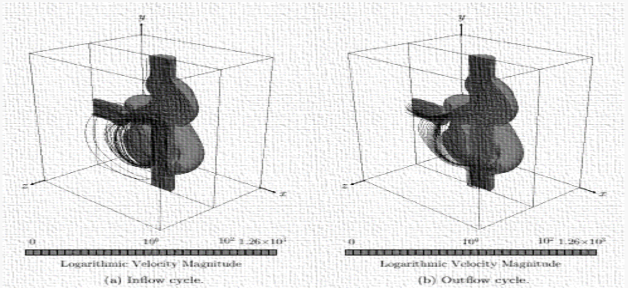
SIGGRAPH 99’



SIGGRAPH 17’

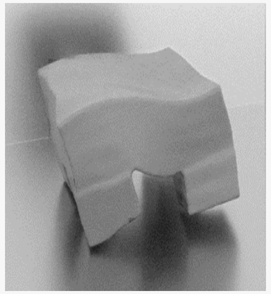
Fluid system optimization

Borrvall and Petersson



Villanueva and Maute

Differentiable physics



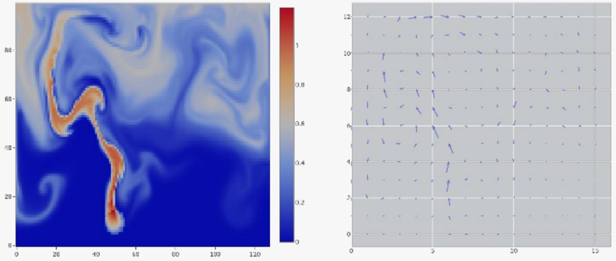
ICRA 19’

PMLR 18’

Related Work

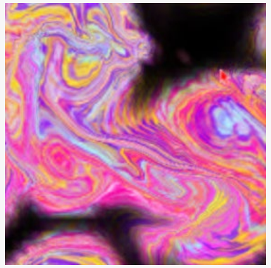
Fluid control

SIGGRAPH 04’

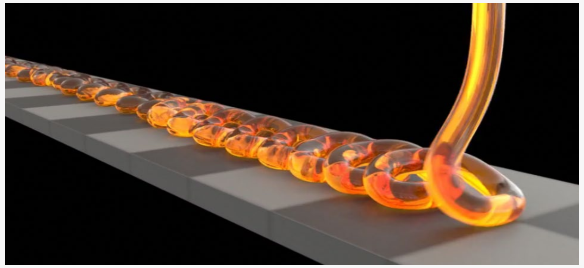


ICLR 20’

Fluid simulation



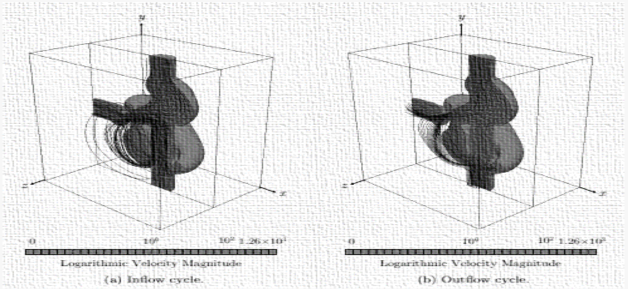
SIGGRAPH 99’



SIGGRAPH 17’

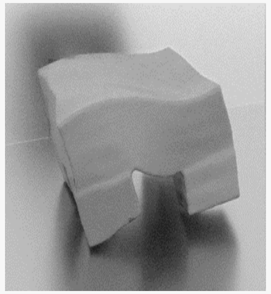
Fluid system optimization

Borrvall and Petersson



Villanueva and Maute

Differentiable physics



ICRA 19’

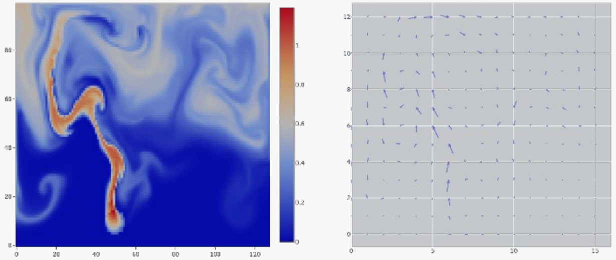
PMLR 18’

Related Work

SIGGRAPH ASIA 2020 VIRTUAL

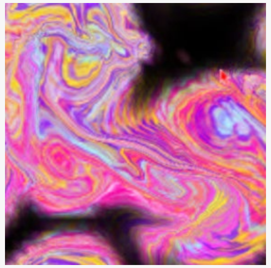
Fluid control

SIGGRAPH 04’

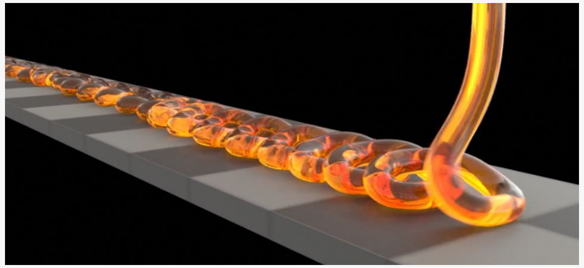


ICLR 20’

Fluid simulation



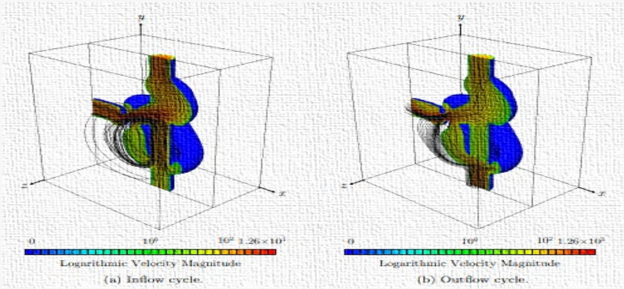
SIGGRAPH 99’



SIGGRAPH 17’

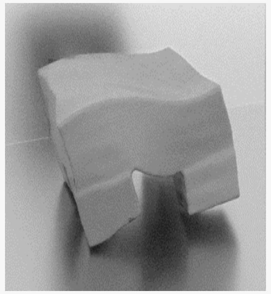
Fluid system optimization

Borrvall and Petersson



Villanueva and Maute

Differentiable physics



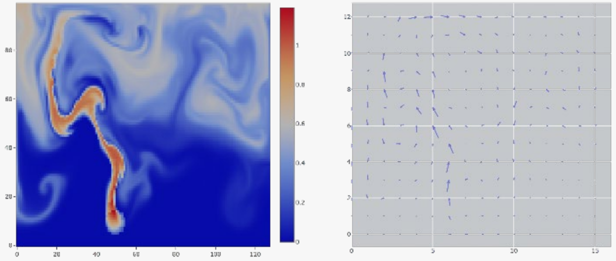
ICRA 19’

PMLR 18’

Related Work

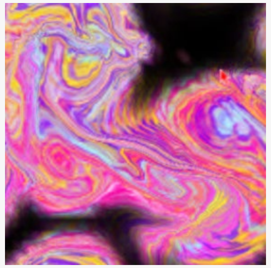
Fluid control

SIGGRAPH 04’

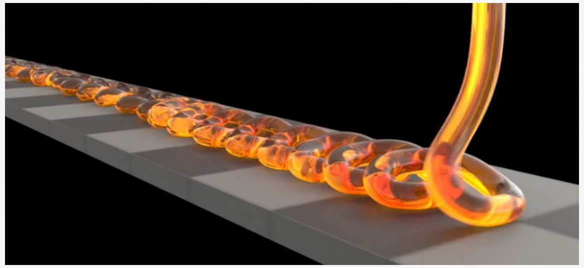


ICLR 20’

Fluid simulation



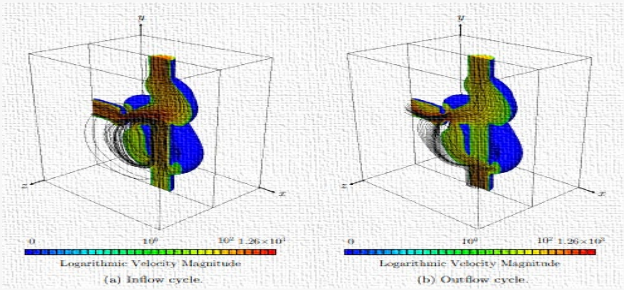
SIGGRAPH 99’



SIGGRAPH 17’

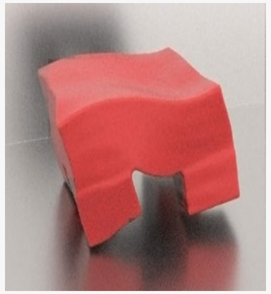
Fluid system optimization

Borrvall and Petersson



Villanueva and Maute

Differentiable physics



ICRA 19’

PMLR 18’

Contributions

Differentiable Stokes flow with a continuous interface

Sub-cell discretization with flexible boundary conditions

Computational design of multi-functional fluidic devices

Contributions

Differentiable Stokes flow with a continuous interface

Sub-cell discretization with flexible boundary conditions

Computational design of multi-functional fluidic devices

Contributions

SIGGRAPH ASIA 2020 VIRTUAL

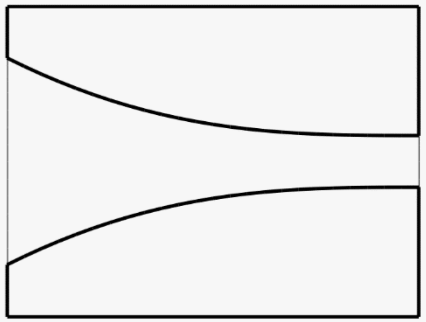
Differentiable Stokes flow with a continuous interface

Sub-cell discretization with flexible boundary conditions

Computational design of multi-functional fluidic devices

Method Overview

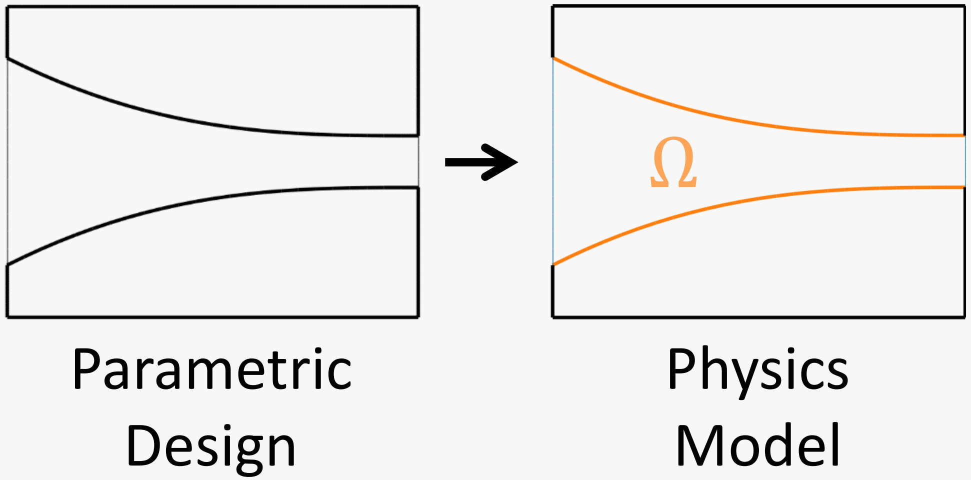
Forward simulation



Parametric Design

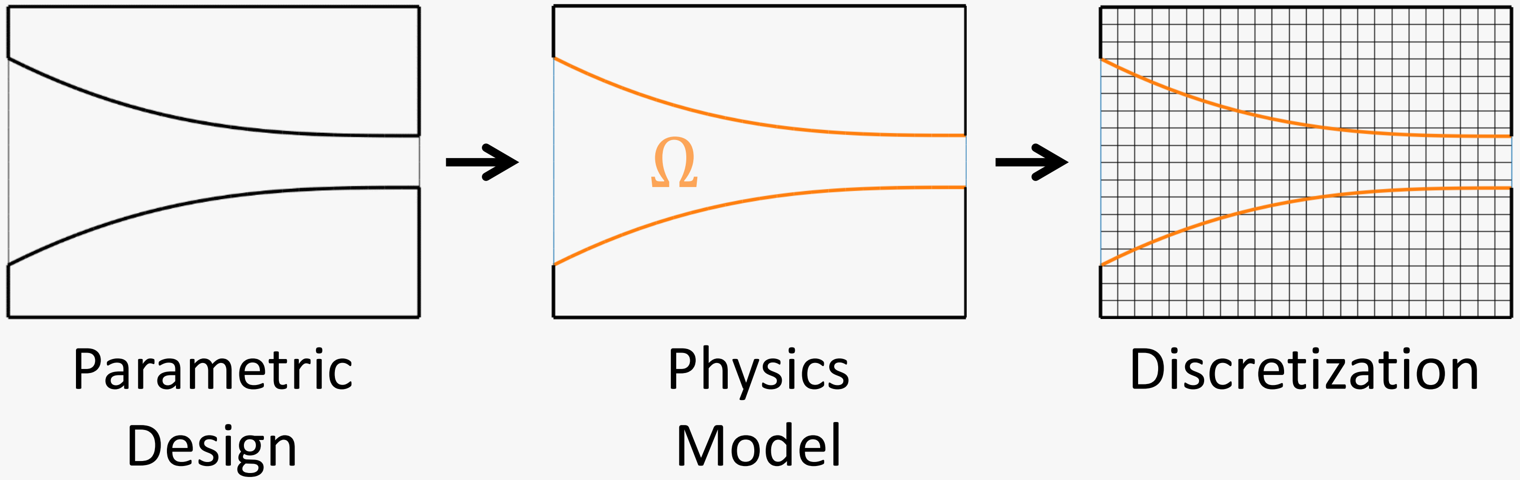
Method Overview

Forward simulation



Method Overview

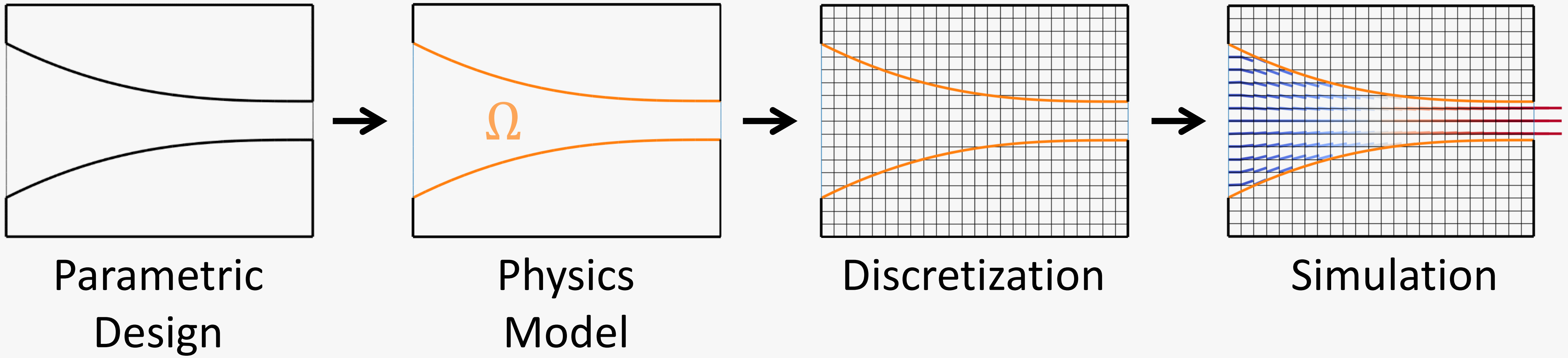
Forward simulation



Method Overview

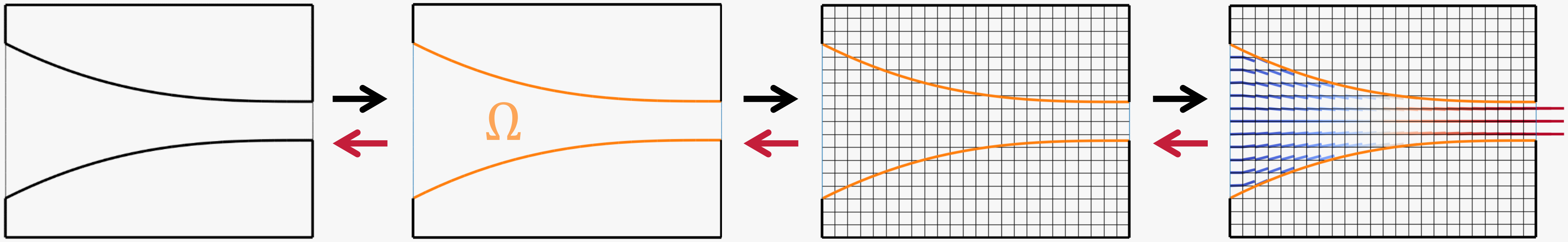
SIGGRAPH ASIA 2020 VIRTUAL

Forward simulation



Method Overview

Forward simulation and backpropagation



Parametric Design (+backprop.)

Physics Model (+backprop.)

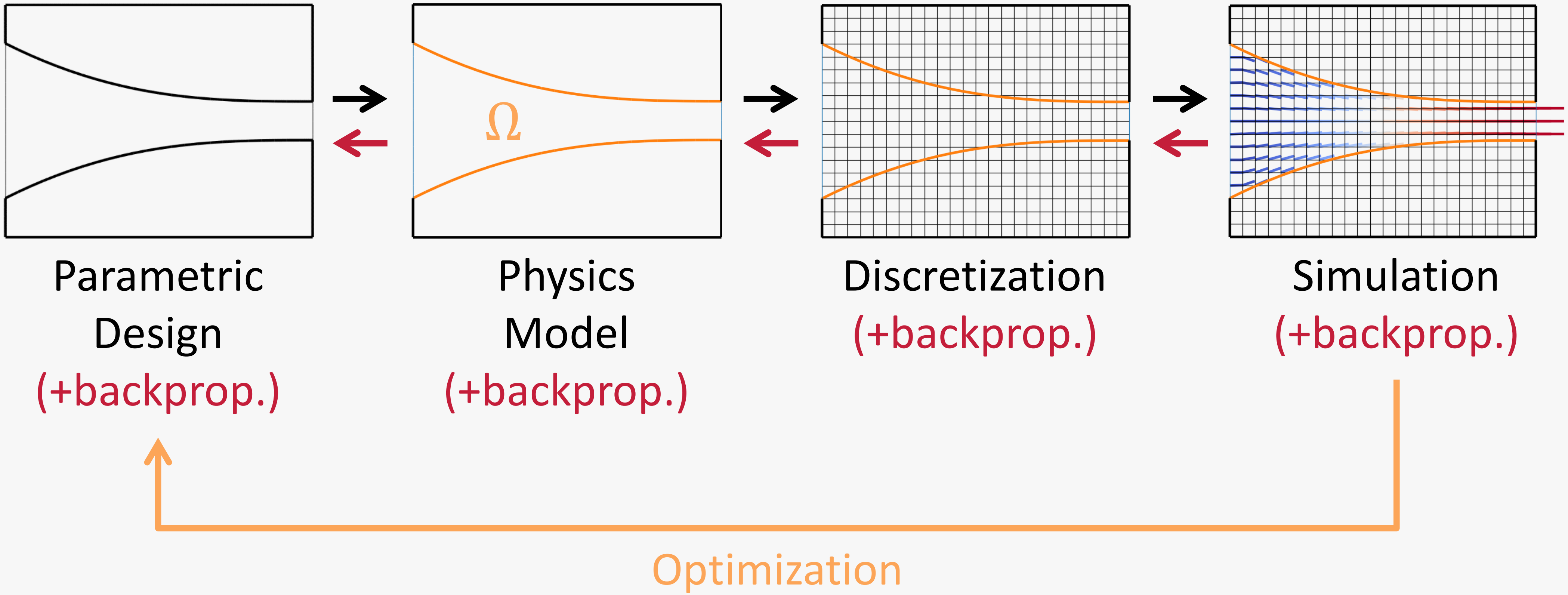
Discretization (+backprop.)

Simulation (+backprop.)

Method Overview

SIGGRAPH ASIA 2020 VIRTUAL

Forward simulation, backpropagation, and optimization



Challenges

Parametrizing the design space (easy)

Simulating the system with a sub-cell discretization (easy)

...and computing gradients

Challenges

Parametrizing the design space (easy)

Simulating the system with a sub-cell discretization (easy)

…and computing gradients

Challenges

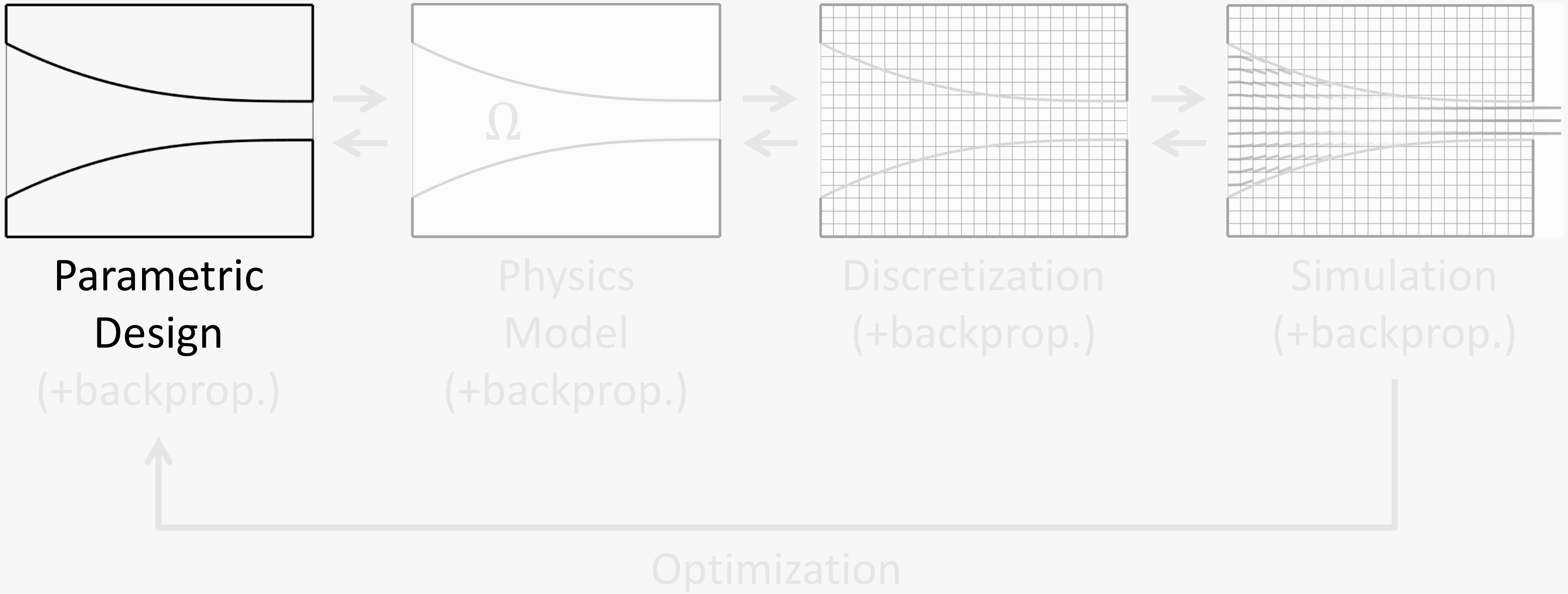
Parametrizing the design space (nontrivial!)

Simulating the system with a sub-cell discretization (nontrivial!)

…and computing gradients (nontrivial!)

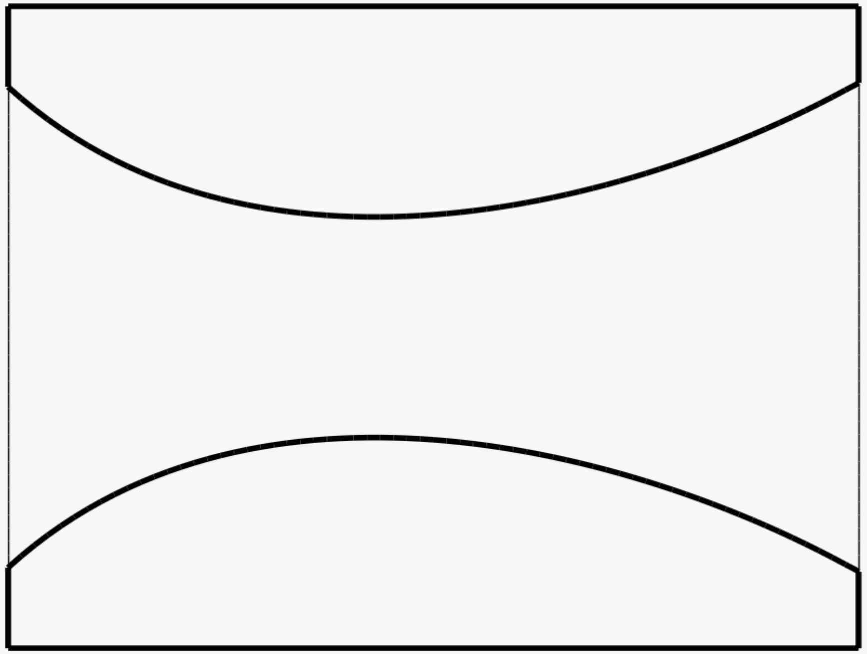
Method: Design Parameters

Forward simulation, backpropagation, and optimization



Method: Design Parameters

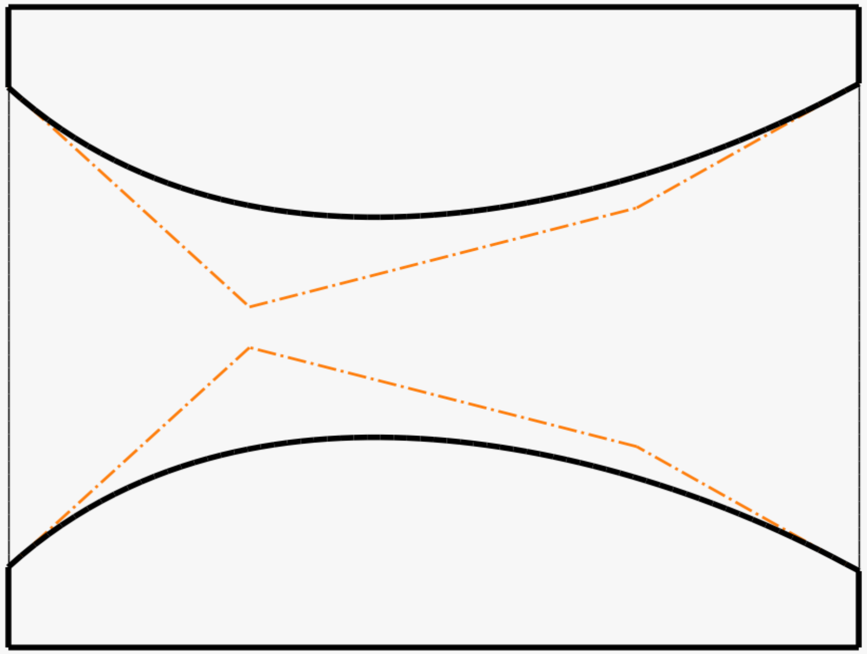
We represent designs as parametric shapes



Method: Design Parameters

SIGGRAPH ASIA 2020 VIRTUAL

We represent designs as parametric shapes



Method: Design Parameters

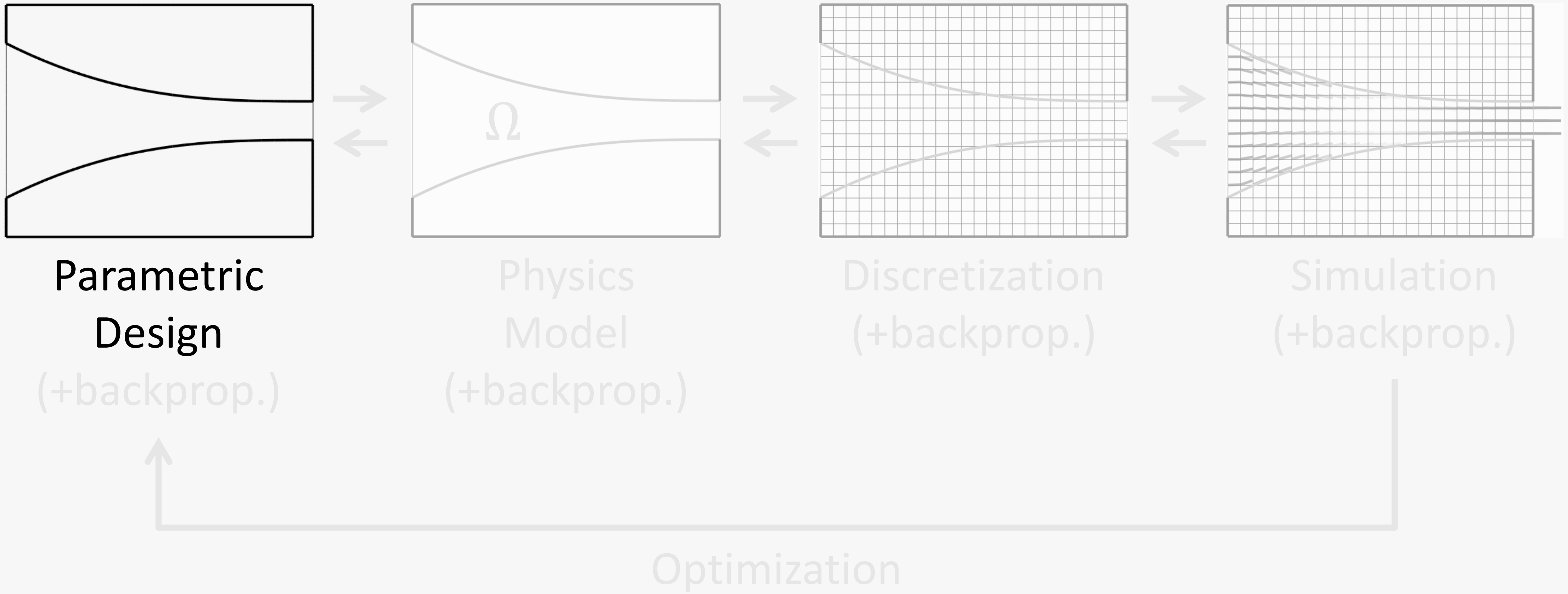
By varying these parameters, we explore different designs

Parametric designs

Signed-distance functions

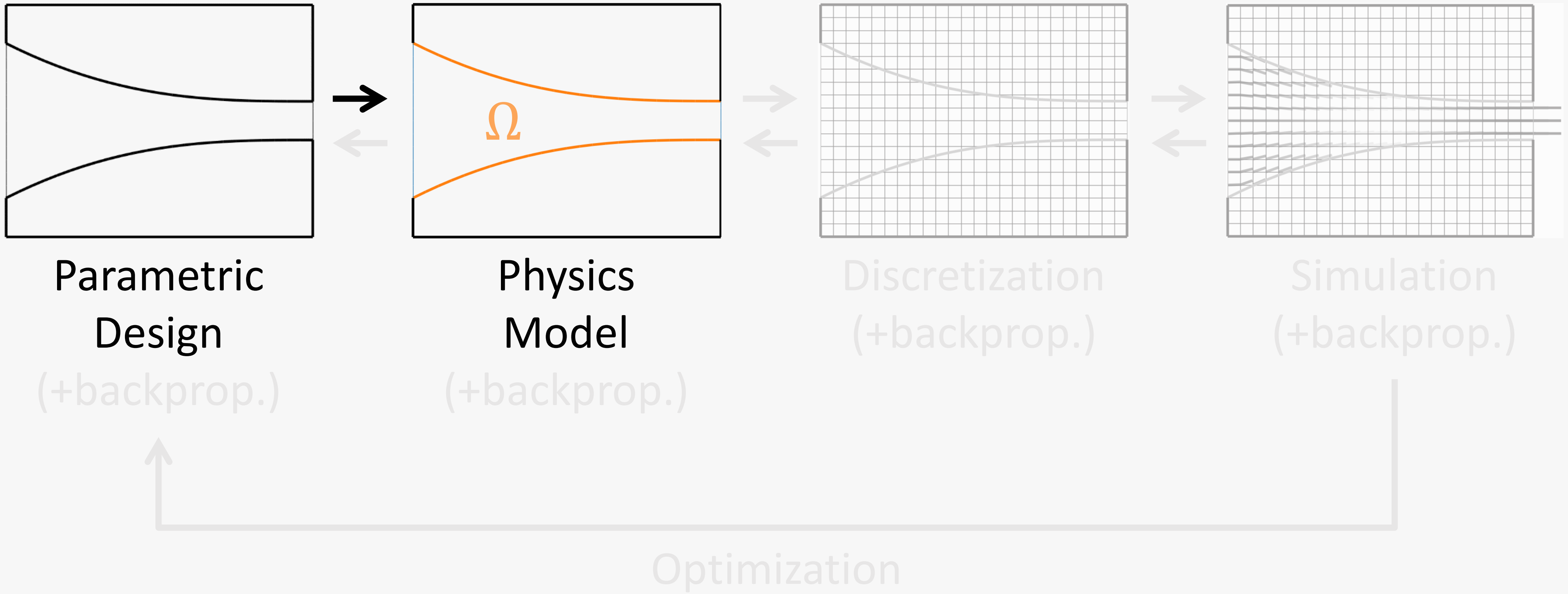
Method: Governing Equations

Forward simulation, backpropagation, and optimization



Method: Governing Equations

Forward simulation, backpropagation, and optimization



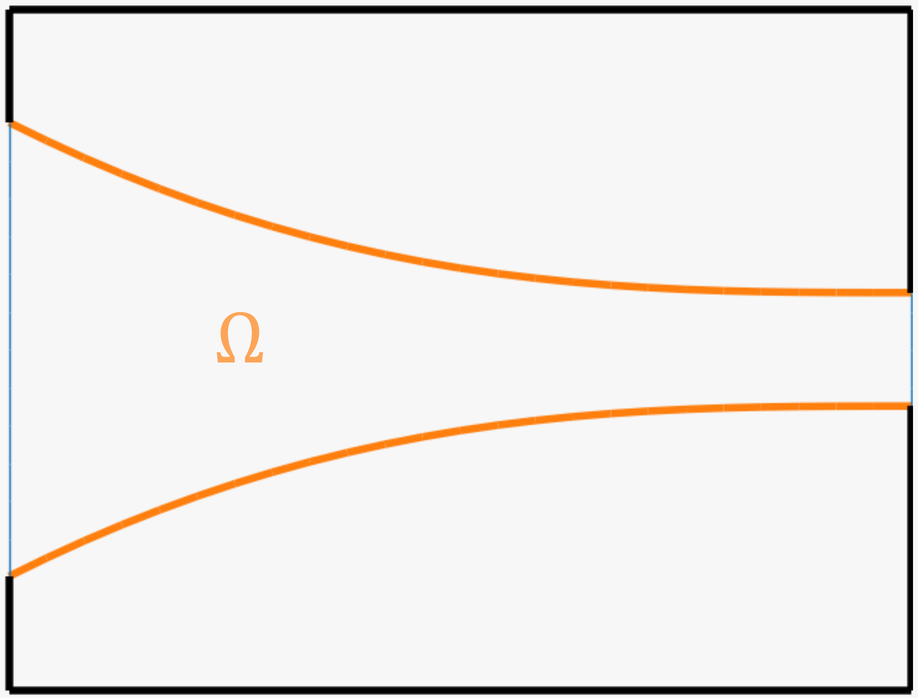
Method: Governing Equations

SIGGRAPH ASIA 2020 VIRTUAL

Incompressible Stokes equations

$$-\eta \Delta \boldsymbol{v}(\boldsymbol{x}) + \nabla p(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega$$

$$\nabla \cdot \boldsymbol{v}(\boldsymbol{x}) = 0, \qquad \boldsymbol{x} \in \Omega$$



η: dynamic viscosity

p: pressure field

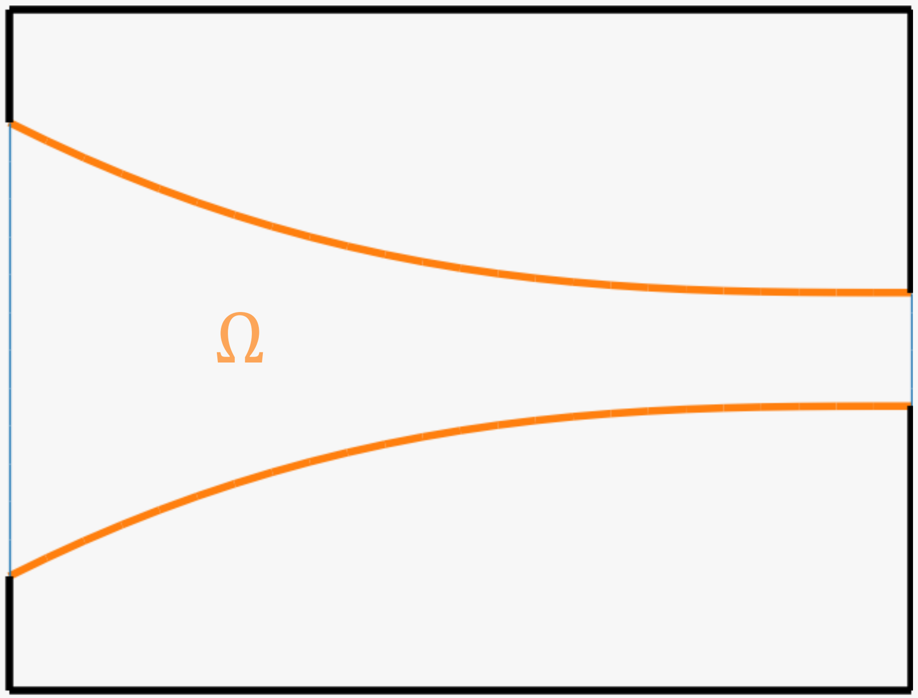
$\boldsymbol{v}$: velocity field

$\boldsymbol{f}$: external force

Method: Governing Equations

Recap: linear elasticity

$$-\mu\Delta\boldsymbol{u}(\boldsymbol{X}) - (\mu + \lambda)\nabla[\nabla \cdot \boldsymbol{u}(\boldsymbol{X})] = \boldsymbol{f}(\boldsymbol{X})$$



μ: Lamé parameters

λ: Lamé parameters

$\boldsymbol{u}$: displacement field

$\boldsymbol{f}$: external force

Method: Governing Equations

SIGGRAPH ASIA 2020 VIRTUAL

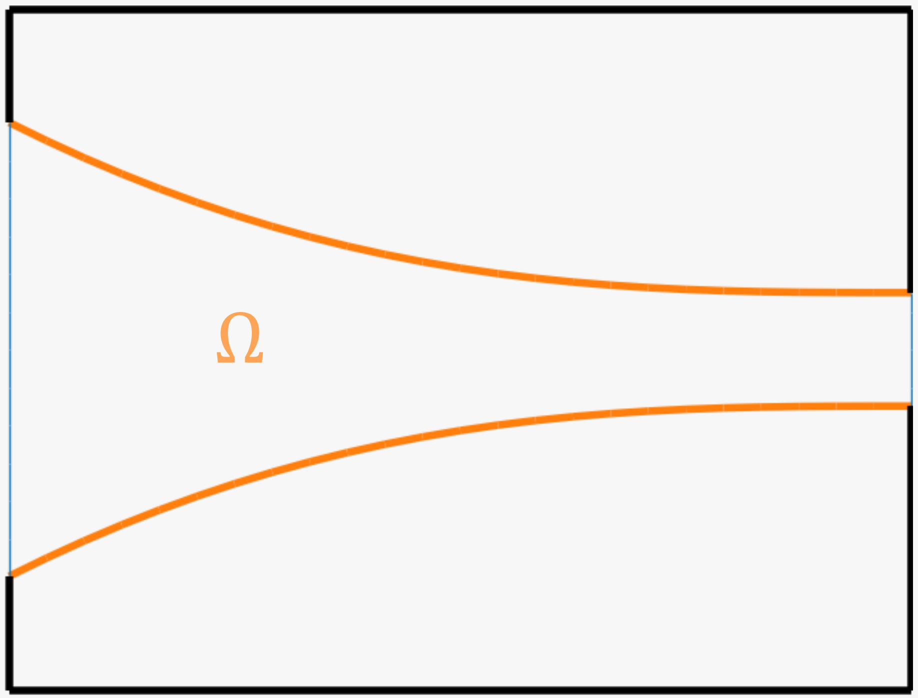
Recap: linear elasticity

$$-\mu\Delta\boldsymbol{u}(\boldsymbol{X}) - (\mu+\lambda)\nabla[\nabla\cdot\boldsymbol{u}(\boldsymbol{X})] = \boldsymbol{f}(\boldsymbol{X})$$

Let $r(\boldsymbol{X}) = -(\mu+\lambda)\nabla\cdot\boldsymbol{u}(\boldsymbol{X})$ and we obtain:

$$-\mu\Delta\boldsymbol{u}(\boldsymbol{X}) + \nabla r(\boldsymbol{X}) = \boldsymbol{f}(\boldsymbol{X}), \qquad \boldsymbol{X}\in\Omega$$

$$\nabla\cdot\boldsymbol{u}(\boldsymbol{X}) + \frac{1}{\mu+\lambda} r(\boldsymbol{X}) = 0, \qquad \boldsymbol{X}\in\Omega$$



Method: Governing Equations

SIGGRAPH ASIA 2020 VIRTUAL

Analogy between Stokes flow and linear elasticity

Stokes flow

$$-\eta\Delta\boldsymbol{v}(\boldsymbol{x}) + \nabla p(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}),\quad \boldsymbol{x}\in\Omega$$

$$\nabla\cdot\boldsymbol{v}(\boldsymbol{x}) = 0,\qquad \boldsymbol{x}\in\Omega$$

Linear elasticity

$$-\mu\Delta\boldsymbol{u}(\boldsymbol{X}) + \nabla r(\boldsymbol{X}) = \boldsymbol{f}(\boldsymbol{X}),\qquad \boldsymbol{X}\in\Omega$$

$$\nabla\cdot\boldsymbol{u}(\boldsymbol{X}) + \frac{1}{\mu+\lambda}r(\boldsymbol{X}) = 0,\qquad \boldsymbol{X}\in\Omega$$

Note the duality between $\eta, \boldsymbol{v}, p$ and $\mu, \boldsymbol{u}, r$.

Right → left when $\lambda\to\infty$ (strict incompressibility).

Method: Governing Equations

Analogy between Stokes flow and linear elasticity

Previous work: use Stokes flow techniques to solve elasticity

Method: Governing Equations

SIGGRAPH ASIA 2020 VIRTUAL

Analogy between Stokes flow and linear elasticity

Previous work: use Stokes flow techniques to solve elasticity

Our model: quasi-incompressible Stokes flow

We use elasticity solvers to solve Stokes flow

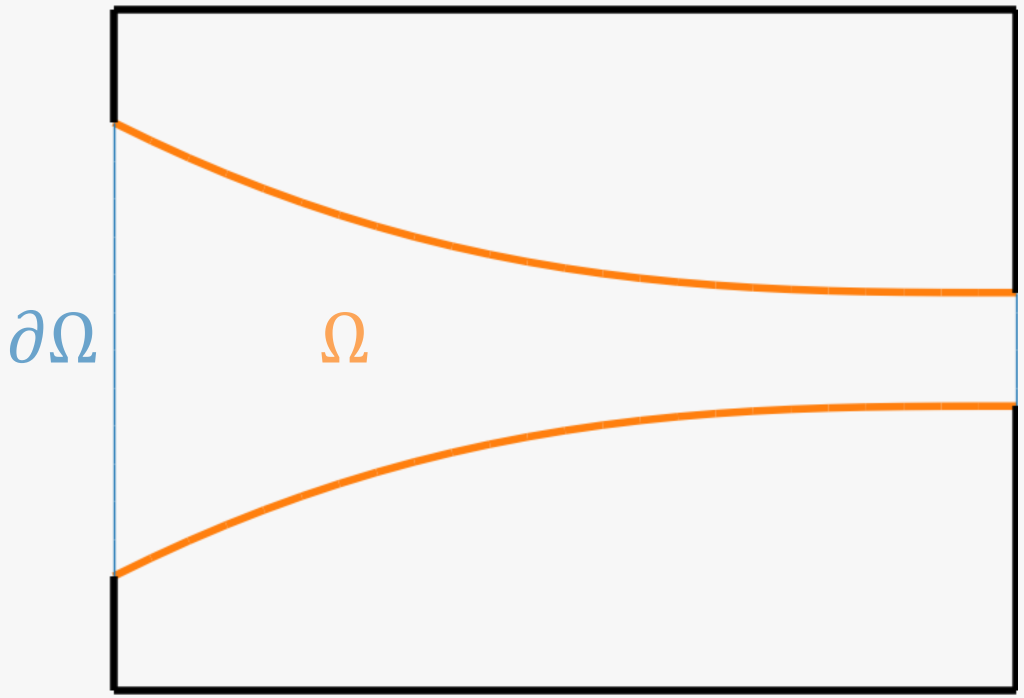
- More numerically robust solvers
- Fewer variables (no pressure term)
- Easier to derive gradients

Method: Governing Equations

A note on boundary conditions: Dirichlet

$$v(x) = \alpha(x), \qquad x \in \partial\Omega$$

α : velocity profile



Method: Governing Equations

A note on boundary conditions: no-slip/no-separation

$$\boldsymbol{v}(\boldsymbol{x}) = \boldsymbol{\alpha}(\boldsymbol{x}), \qquad \boldsymbol{x} \in \partial\Omega$$

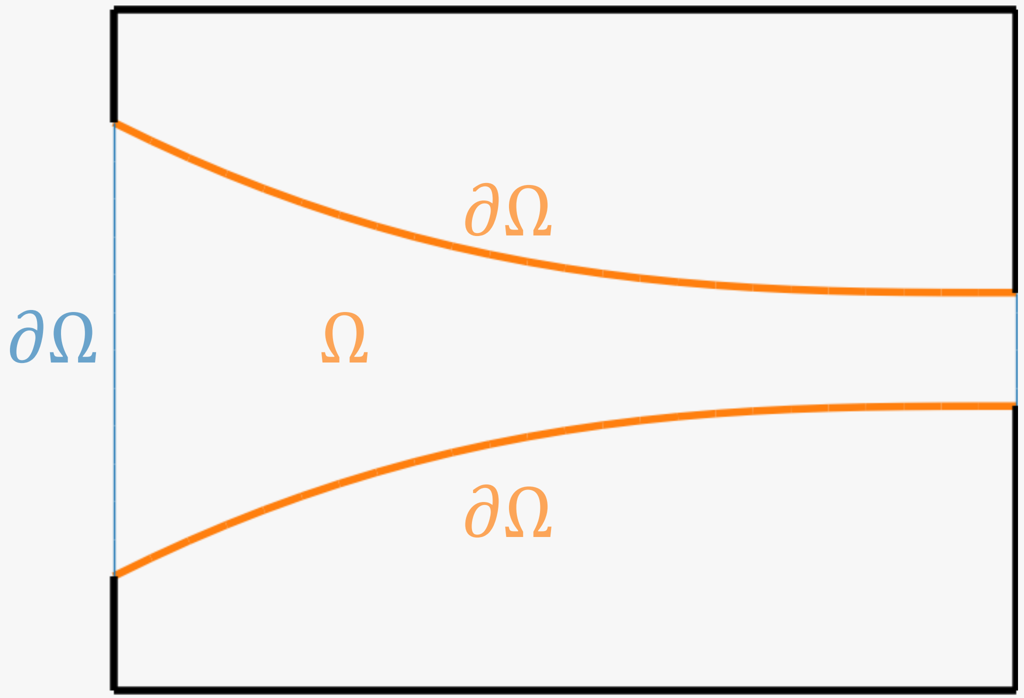
$$\boldsymbol{v}(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x}) = 0, \qquad \boldsymbol{x} \in \partial\Omega$$

$$\boldsymbol{\tau}_t(\boldsymbol{x}) = \boldsymbol{0}, \qquad \boldsymbol{x} \in \partial\Omega$$

$\boldsymbol{\alpha}$: velocity profile

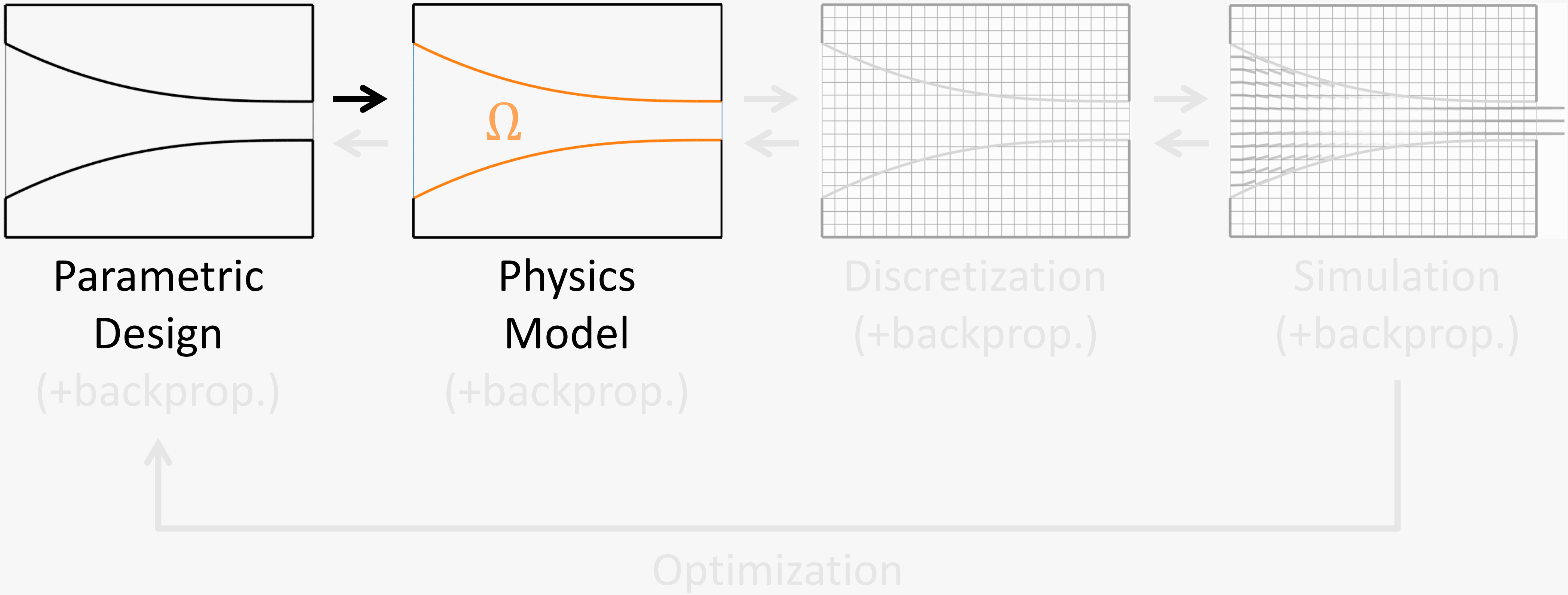
$\boldsymbol{n}$: normal

$\boldsymbol{\tau}_t$: tangent traction



Method: Discretization

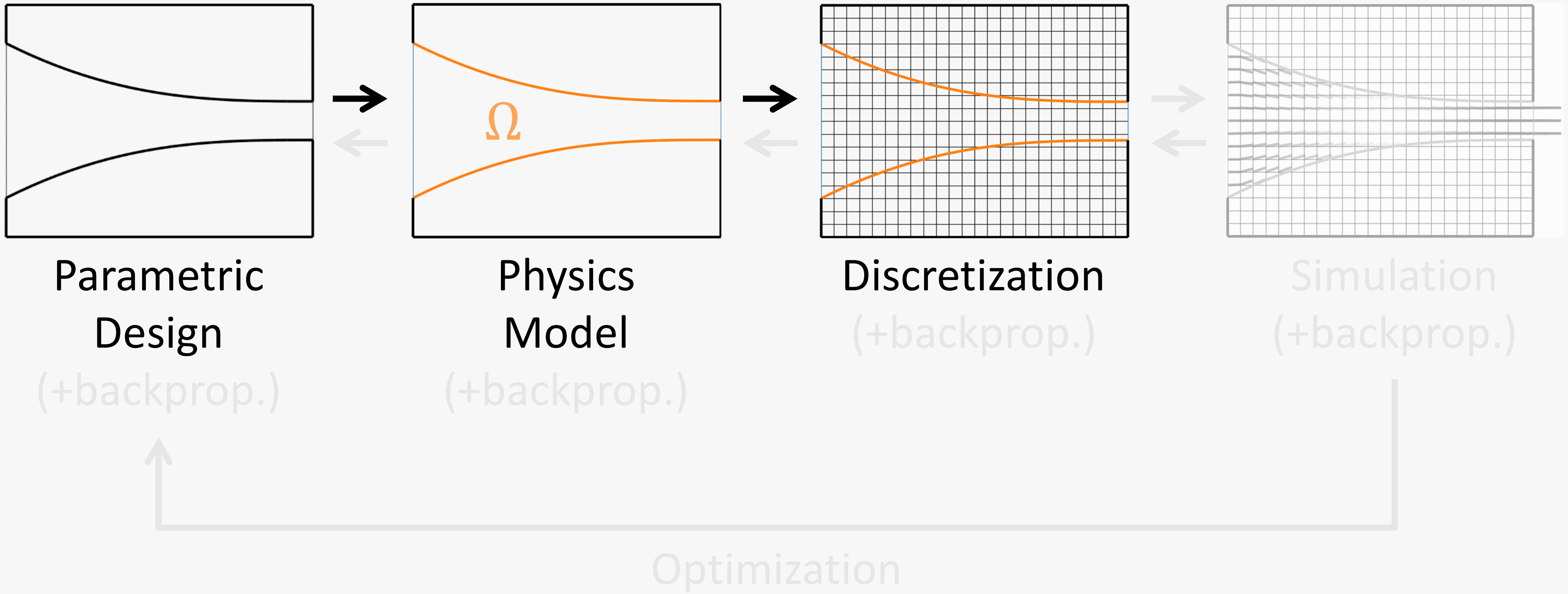
Forward simulation, backpropagation, and optimization



Method: Discretization

SIGGRAPH ASIA 2020 VIRTUAL

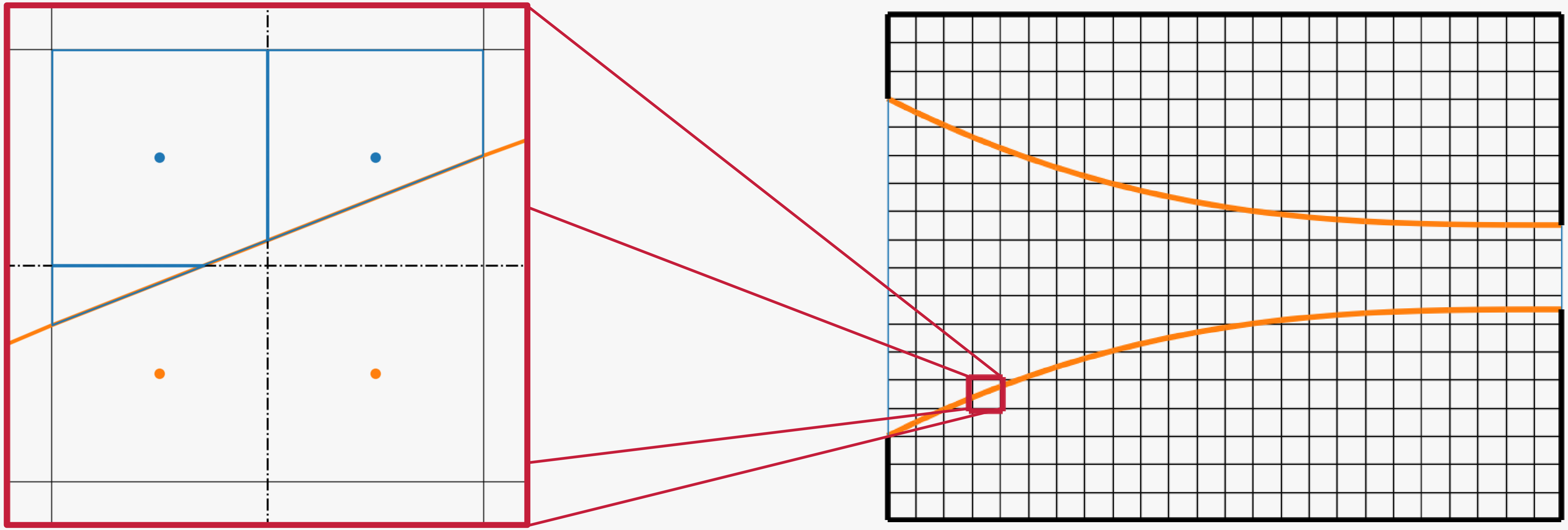
Forward simulation, backpropagation, and optimization



Method: Discretization

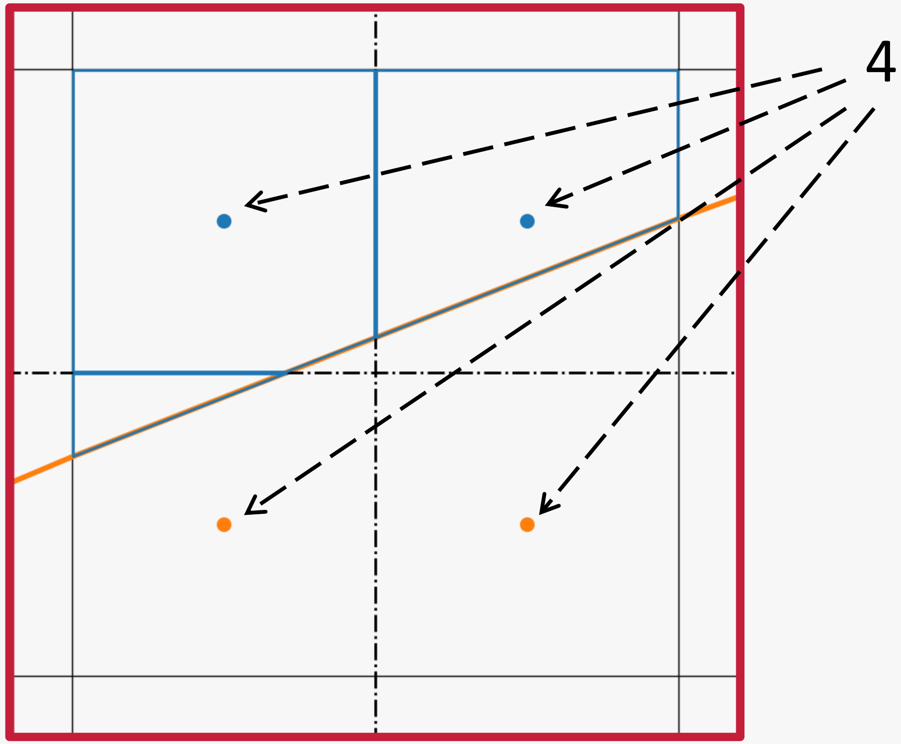
SIGGRAPH ASIA 2020 VIRTUAL

Consider a hybrid cell



Method: Discretization

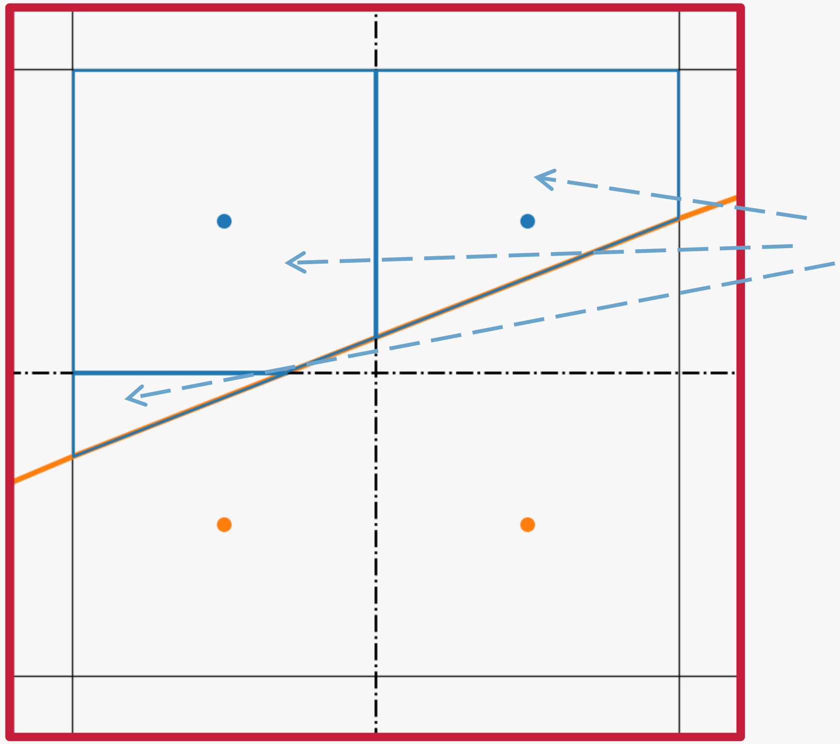
Consider a hybrid cell



4 Gaussian quadrature points in each cell

Method: Discretization

Consider a hybrid cell

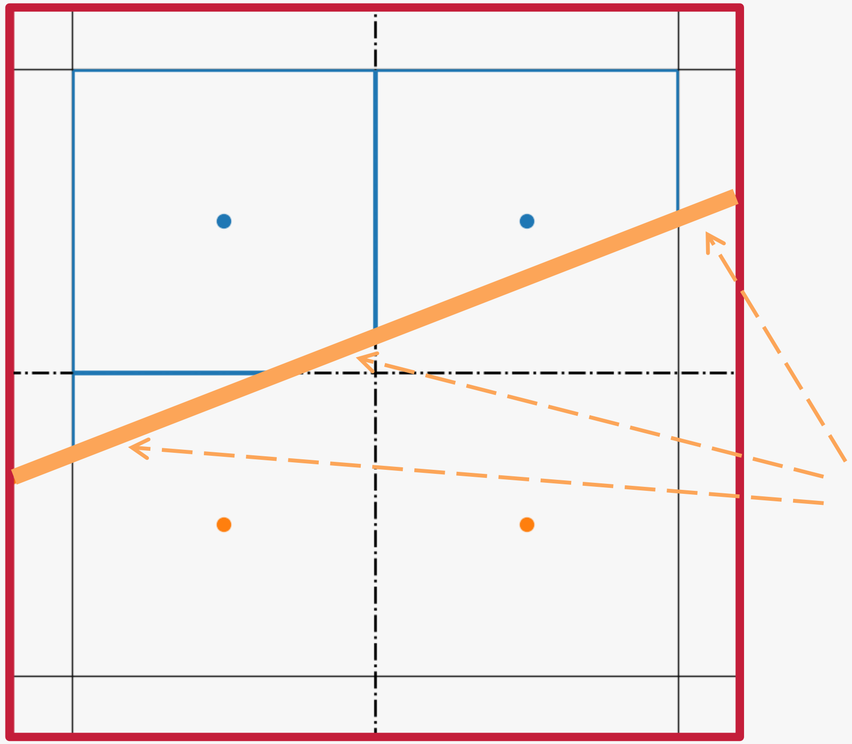


4 Gaussian quadrature points in each cell

Weight of each point = area of the polygon

Method: Discretization

Consider a hybrid cell



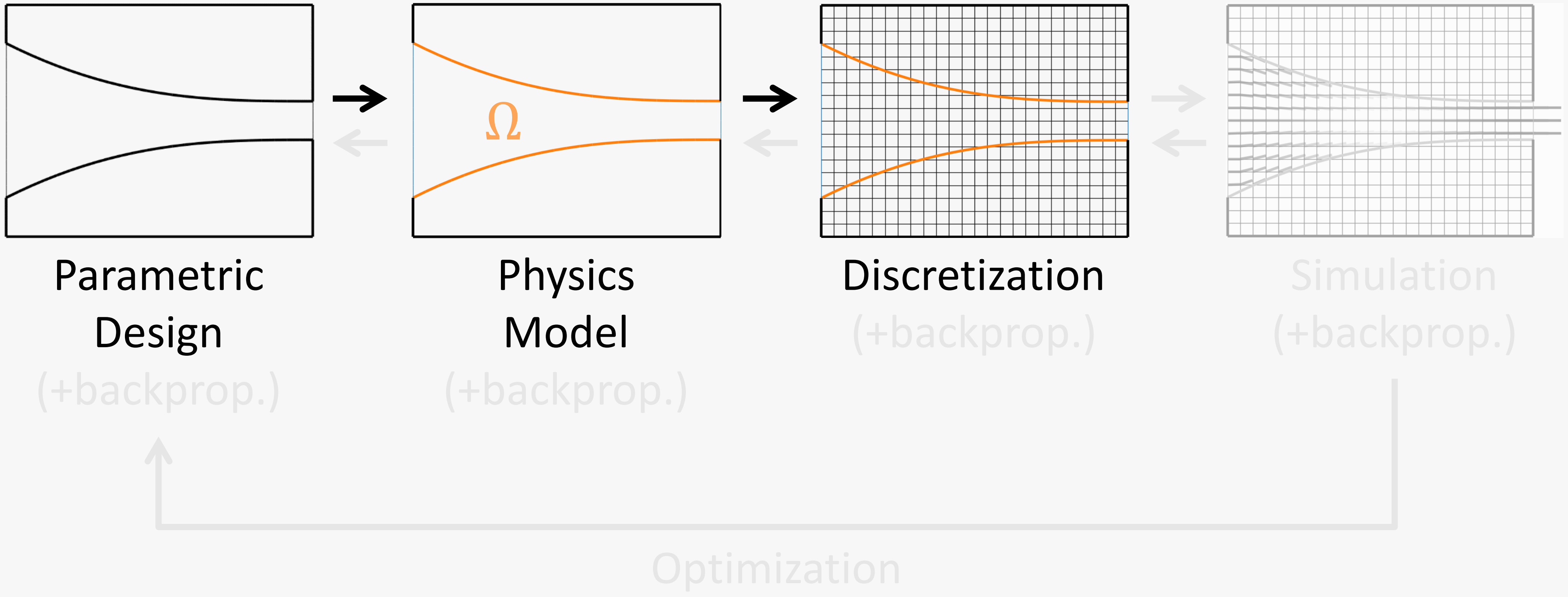
4 Gaussian quadrature points in each cell

Weight of each point = area of the polygon

Boundary conditions are integrated along the interface

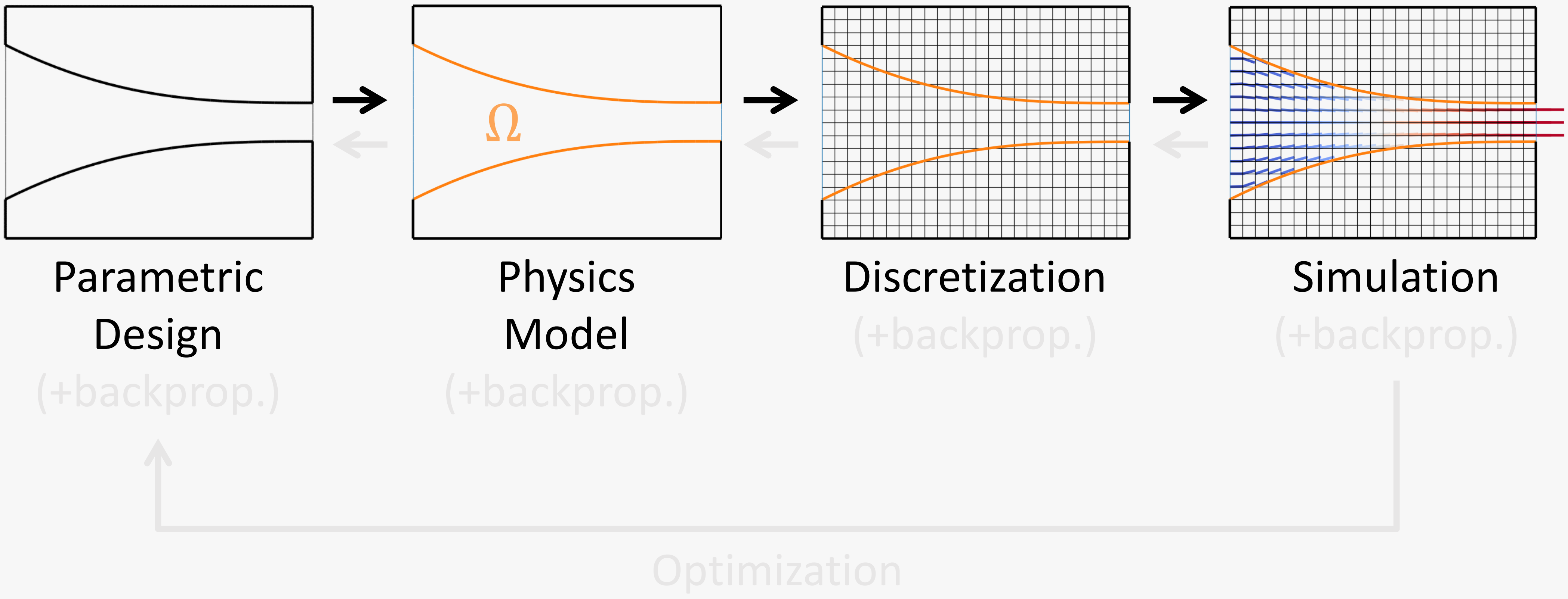
Method: Simulation

Forward simulation, backpropagation, and optimization



Method: Simulation

Forward simulation, backpropagation, and optimization



Method: Simulation

Recap: quasi-incompressible Stokes flow (linear elasticity)

$$-\mu\Delta\boldsymbol{u}(\boldsymbol{X}) - (\mu+\lambda)\nabla[\nabla\cdot\boldsymbol{u}(\boldsymbol{X})] = \boldsymbol{f}(\boldsymbol{X})$$

$$s.t.\quad \text{Boundary conditions.}$$

Method: Simulation

Recap: quasi-incompressible Stokes flow (linear elasticity)

$$-\mu\Delta\boldsymbol{u}(\boldsymbol{X}) - (\mu+\lambda)\nabla[\nabla\cdot\boldsymbol{u}(\boldsymbol{X})] = \boldsymbol{f}(\boldsymbol{X})$$

$$s.t. \quad \text{Boundary conditions.}$$

After discretization from the variational form (Quadratic programming)

$$\min_{\boldsymbol{u}} \boldsymbol{u}^\top \boldsymbol{K}(\boldsymbol{\theta})\boldsymbol{u}$$

$$s.t.\, \boldsymbol{C}(\boldsymbol{\theta})\boldsymbol{u} = \boldsymbol{d}(\boldsymbol{\theta})$$

Method: Simulation

Recap: quasi-incompressible Stokes flow (linear elasticity)

$$-\mu\Delta\boldsymbol{u}(\boldsymbol{X}) - (\mu+\lambda)\nabla[\nabla\cdot\boldsymbol{u}(\boldsymbol{X})] = \boldsymbol{f}(\boldsymbol{X})$$

$$s.t. \quad \text{Boundary conditions.}$$

After discretization from the variational form (Quadratic programming)

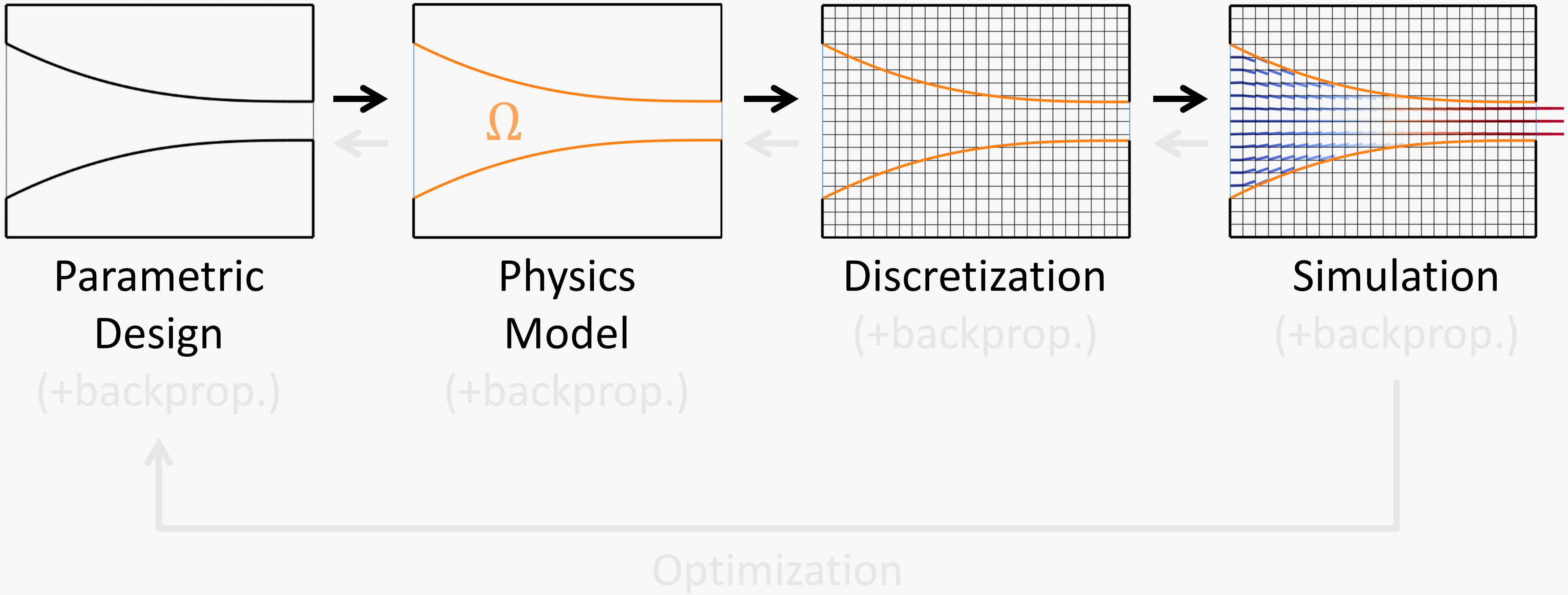
$$\min_{\boldsymbol{u}} \boldsymbol{u}^\top \boldsymbol{K}(\boldsymbol{\theta})\boldsymbol{u}$$

$$s.t. \; \boldsymbol{C}(\boldsymbol{\theta})\boldsymbol{u} = \boldsymbol{d}(\boldsymbol{\theta})$$

Note that the stiffness matrix and the boundary conditions are determined by the design parameter $\boldsymbol{\theta}$.

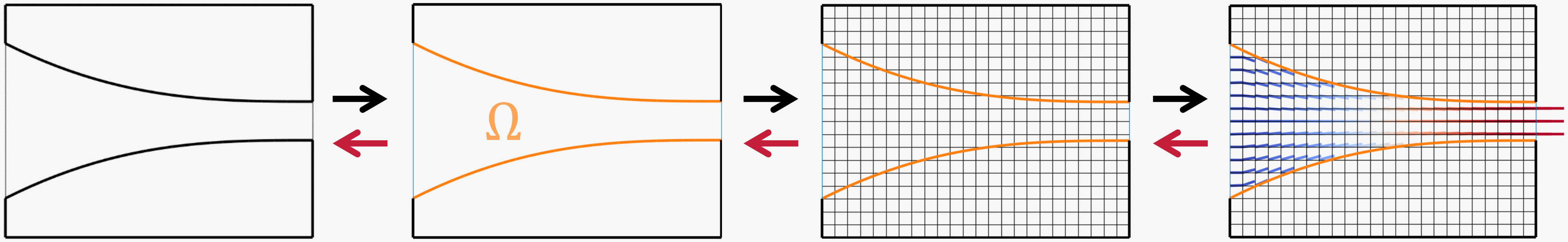
Recap: Forward Simulation

Forward simulation, backpropagation, and optimization



Method: Backpropagation

Forward simulation, backpropagation, and optimization



Parametric Design (+backprop.)

Physics Model (+backprop.)

Discretization (+backprop.)

Simulation (+backprop.)

Optimization

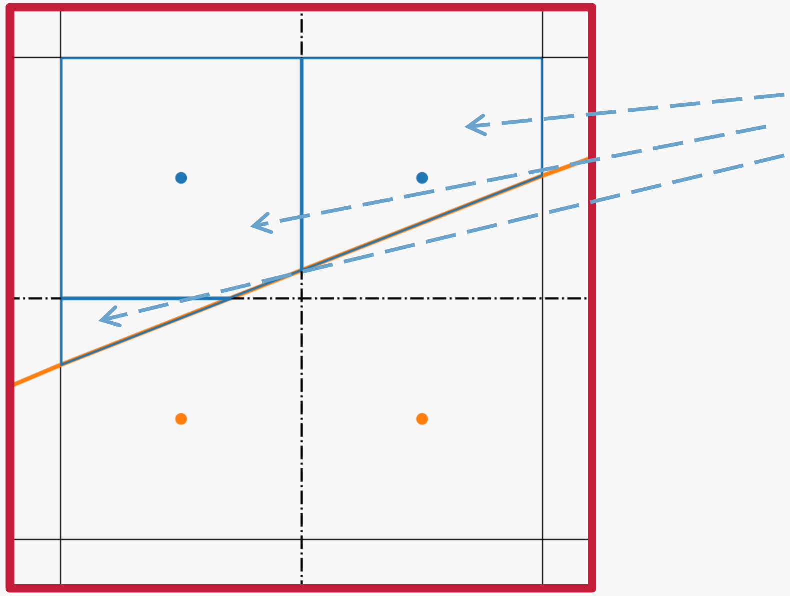
Method: Backpropagation

SIGGRAPH ASIA 2020 VIRTUAL

Most of the computation requires the chain rule only

But there are two exceptions!

Exception 1: gradients w.r.t. the area of a polygon



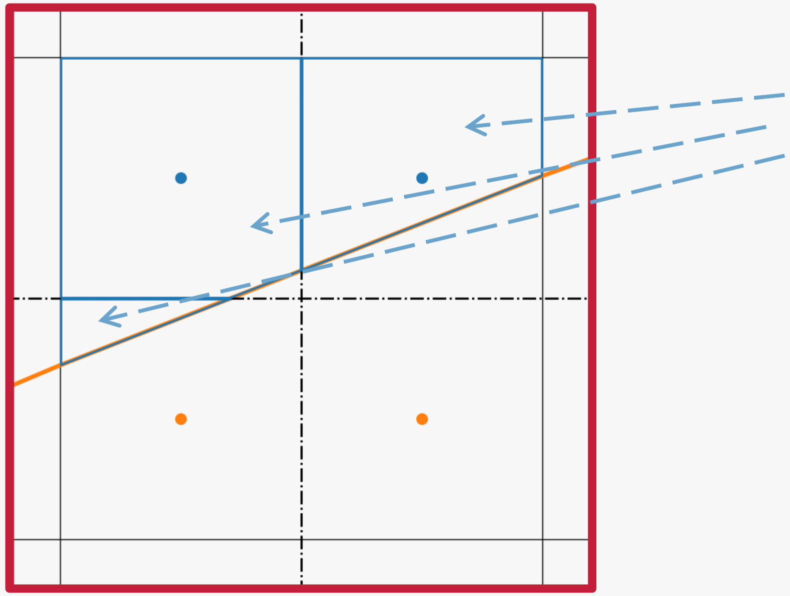
A brute-force implementation plus autodiff leads to lots of if-else branches!

Method: Backpropagation

Most of the computation requires the chain rule only

But there are two exceptions!

Exception 1: gradients w.r.t. the area of a polygon



A brute-force implementation plus autodiff leads to lots of if-else branches!

Our solution: deriving gradients from a closed-form solution [Barrow 79']

Method: Backpropagation

Most of the computation requires the chain rule only

But there are two exceptions!

Exception 2: gradients through the QP problem

$$\min_{\boldsymbol{u}} \boldsymbol{u}^\top \boldsymbol{K}(\boldsymbol{\theta})\boldsymbol{u}$$

$$s.t.\ \boldsymbol{C}(\boldsymbol{\theta})\boldsymbol{u} = \boldsymbol{d}(\boldsymbol{\theta})$$

Method: Backpropagation

SIGGRAPH ASIA 2020 VIRTUAL

Most of the computation requires the chain rule only

But there are two exceptions!

Exception 2: gradients through the QP problem

$$\min_{\boldsymbol{u}} \boldsymbol{u}^\top \boldsymbol{K}(\boldsymbol{\theta})\boldsymbol{u} \quad s.t.\ \boldsymbol{C}(\boldsymbol{\theta})\boldsymbol{u} = \boldsymbol{d}(\boldsymbol{\theta})$$

$$\longleftrightarrow \begin{pmatrix} \boldsymbol{K}(\boldsymbol{\theta}) & \boldsymbol{C}^\top(\boldsymbol{\theta}) \\ \boldsymbol{C}(\boldsymbol{\theta}) & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{u}} \\ \tilde{\boldsymbol{\lambda}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{d}(\boldsymbol{\theta}) \end{pmatrix}$$

KKT conditions

Method: Backpropagation

SIGGRAPH ASIA 2020 VIRTUAL

Most of the computation requires the chain rule only

But there are two exceptions!

Exception 2: gradients through the QP problem (matrix reused)

$$\min_{\boldsymbol{u}} \boldsymbol{u}^\top \boldsymbol{K}(\boldsymbol{\theta})\boldsymbol{u} \quad s.t.\ \boldsymbol{C}(\boldsymbol{\theta})\boldsymbol{u} = \boldsymbol{d}(\boldsymbol{\theta})$$

$\leftrightarrow$

$$\begin{pmatrix} \boldsymbol{K}(\boldsymbol{\theta}) & \boldsymbol{C}^\top(\boldsymbol{\theta}) \\ \boldsymbol{C}(\boldsymbol{\theta}) & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{u}} \\ \tilde{\boldsymbol{\lambda}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{d}(\boldsymbol{\theta}) \end{pmatrix}$$

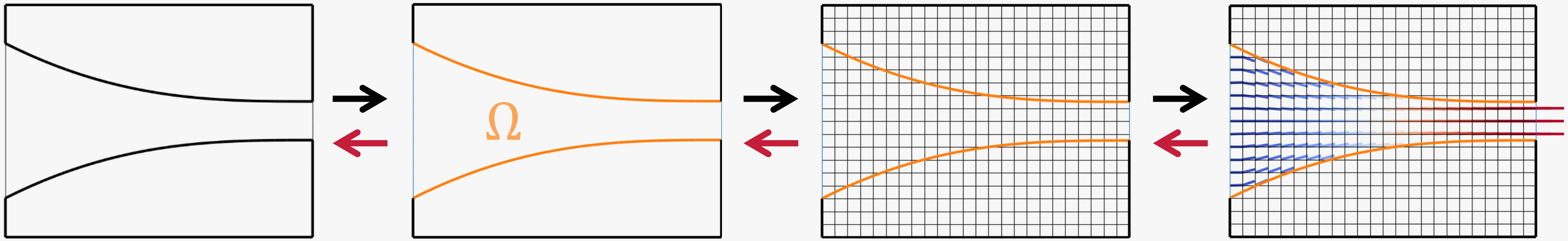
KKT conditions

$$\begin{pmatrix} \boldsymbol{K}(\boldsymbol{\theta}) & \boldsymbol{C}^\top(\boldsymbol{\theta}) \\ \boldsymbol{C}(\boldsymbol{\theta}) & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \delta\tilde{\boldsymbol{u}} \\ \delta\tilde{\boldsymbol{\lambda}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \delta\boldsymbol{d}(\boldsymbol{\theta}) \end{pmatrix} - \begin{pmatrix} \delta\boldsymbol{K}(\boldsymbol{\theta}) & \delta\boldsymbol{C}^\top(\boldsymbol{\theta}) \\ \delta\boldsymbol{C}(\boldsymbol{\theta}) & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{u}} \\ \tilde{\boldsymbol{\lambda}} \end{pmatrix}$$

Sensitivity analysis

Method: Optimization

Forward simulation, backpropagation, and optimization



Parametric Design (+backprop.)

Physics Model (+backprop.)

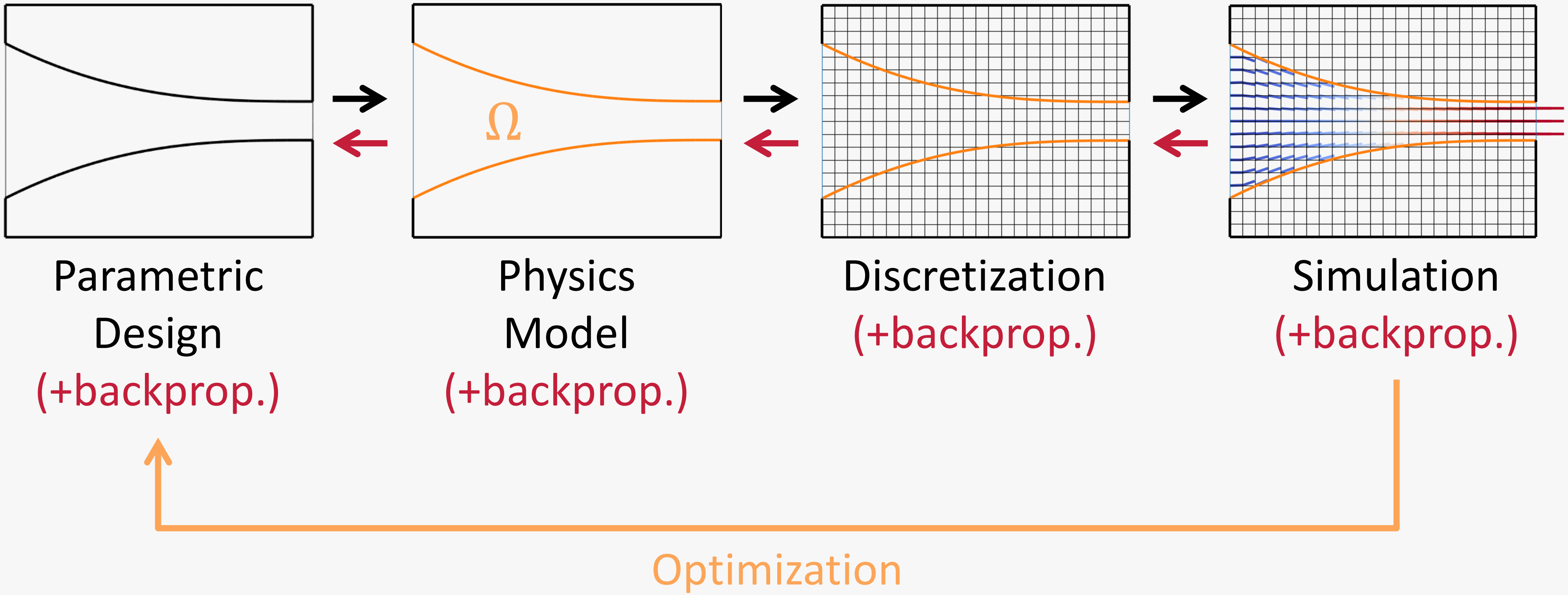
Discretization (+backprop.)

Simulation (+backprop.)

Optimization

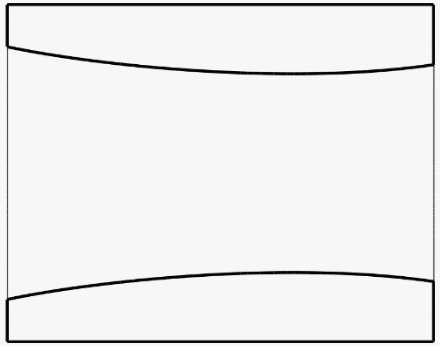
Method: Optimization

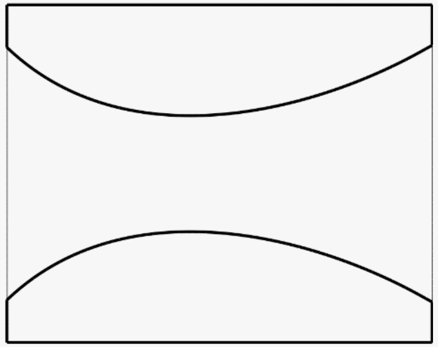
Forward simulation, backpropagation, and optimization

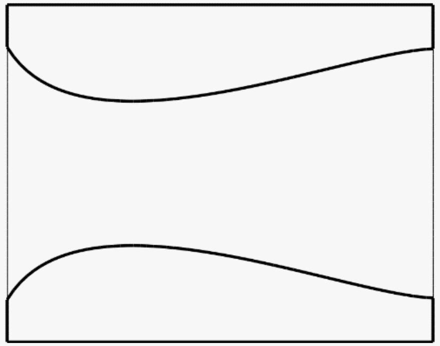


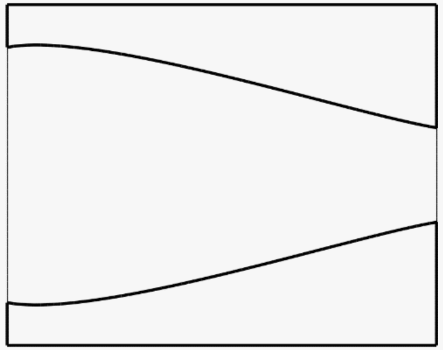
Method: Optimization

Sample a few random designs


Design #1

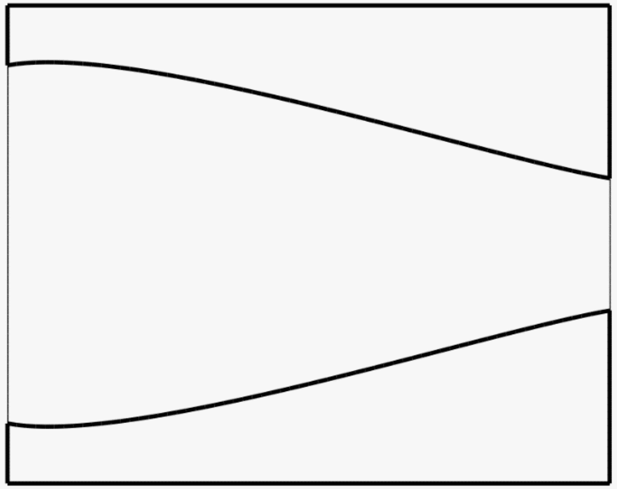

Design #2


Design #3


Design #4

Method: Optimization

Pick the best one to initialize the optimization

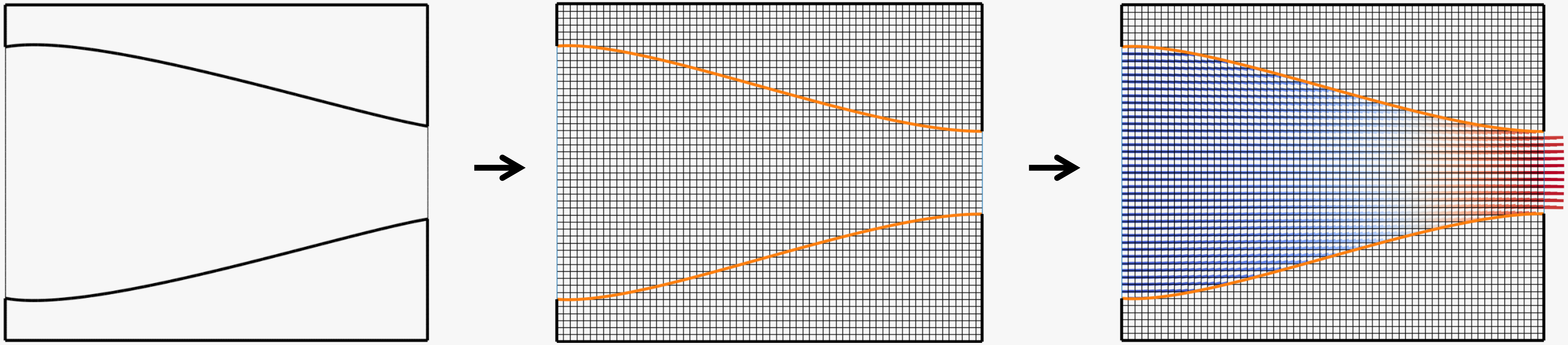


Best initial guess

L-BFGS optimization

Method: Optimization

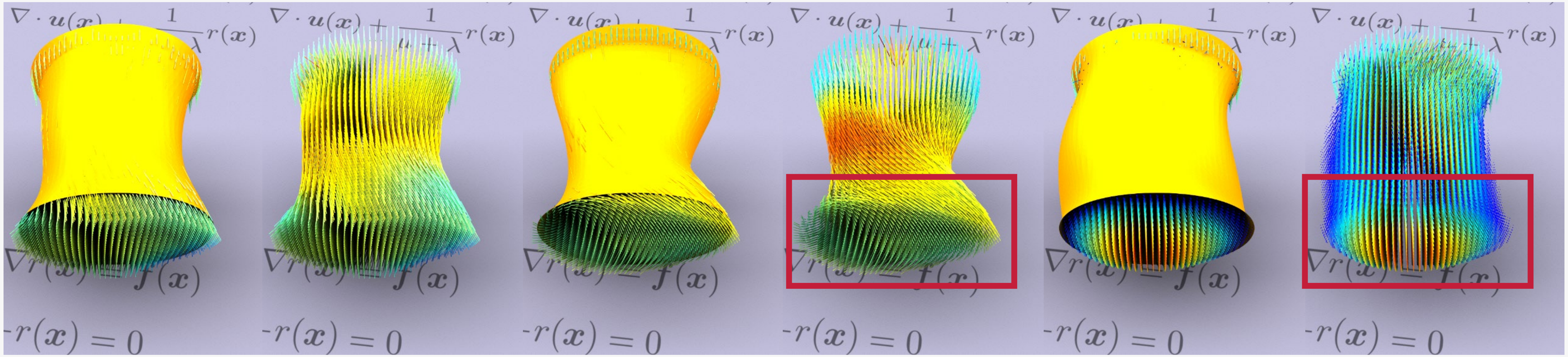
Pick the best one to initialize the optimization



Best initial guess — L-BFGS optimization — Optimal solution

Results: Fluidic Twister

Flexible handling of boundary conditions matters



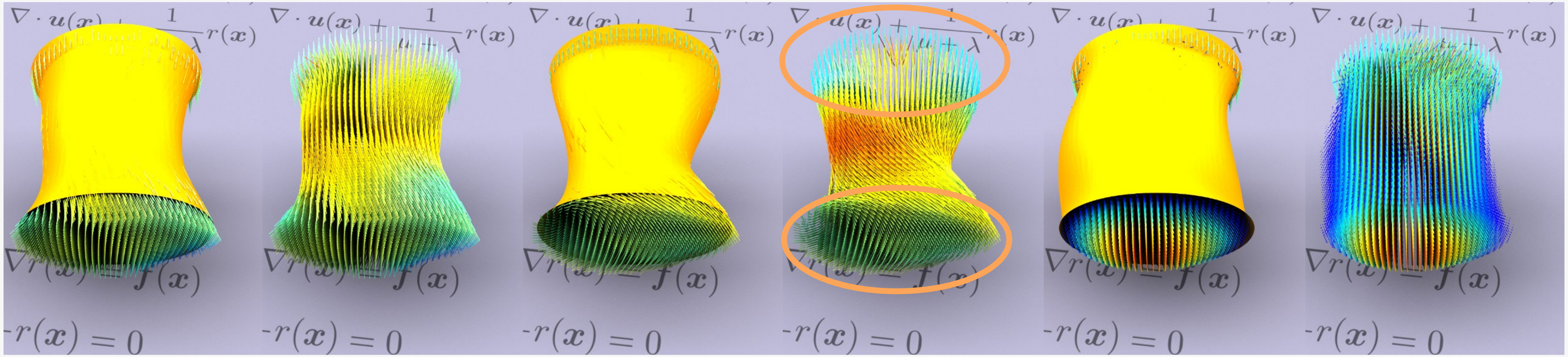
Initial guess

Optimized design
(no-separation)

Optimized design
(no-slip)

Results: Fluidic Twister

Flexible handling of boundary conditions matters



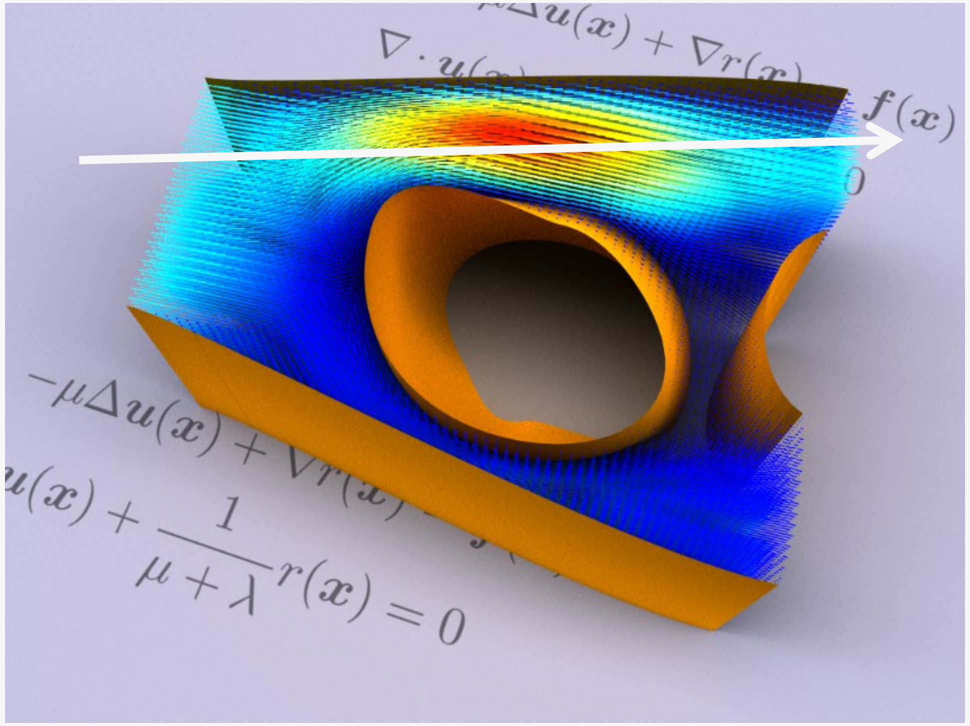
Initial guess

Optimized design (no-separation)

Optimized design (no-slip)

Results: Fluidic Switch

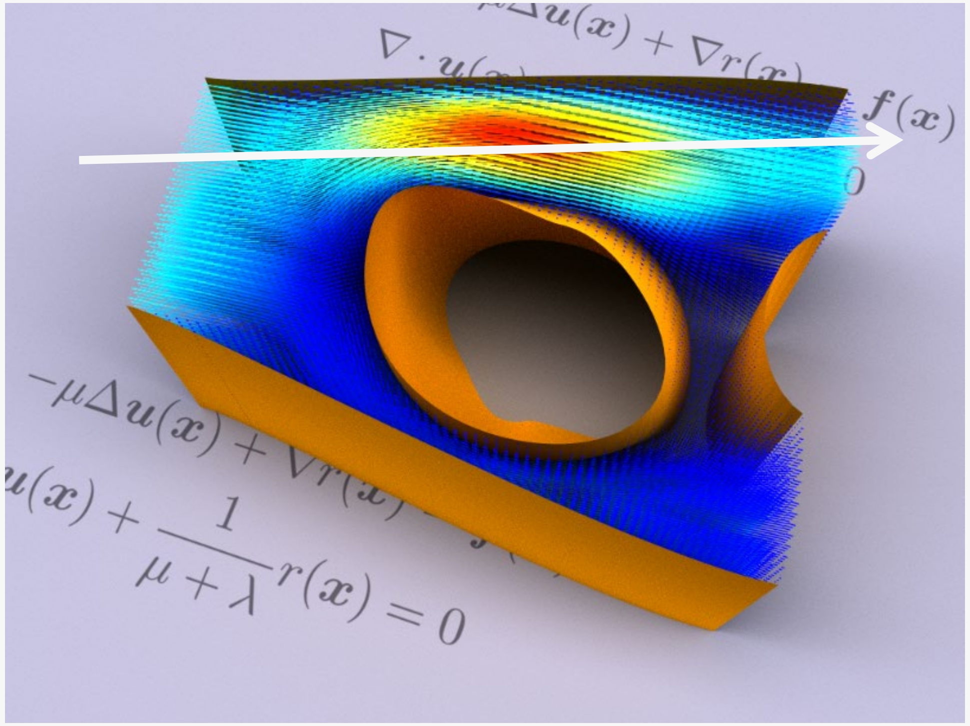
Optimization with multiple configurations



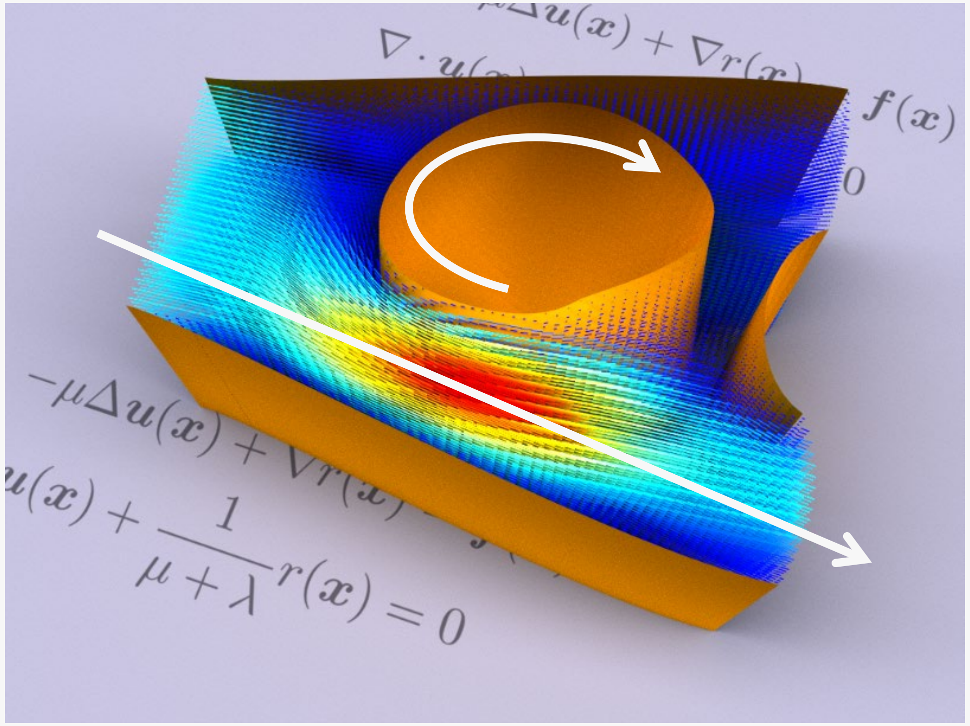
Switch is off

Results: Fluidic Switch

Optimization with multiple configurations



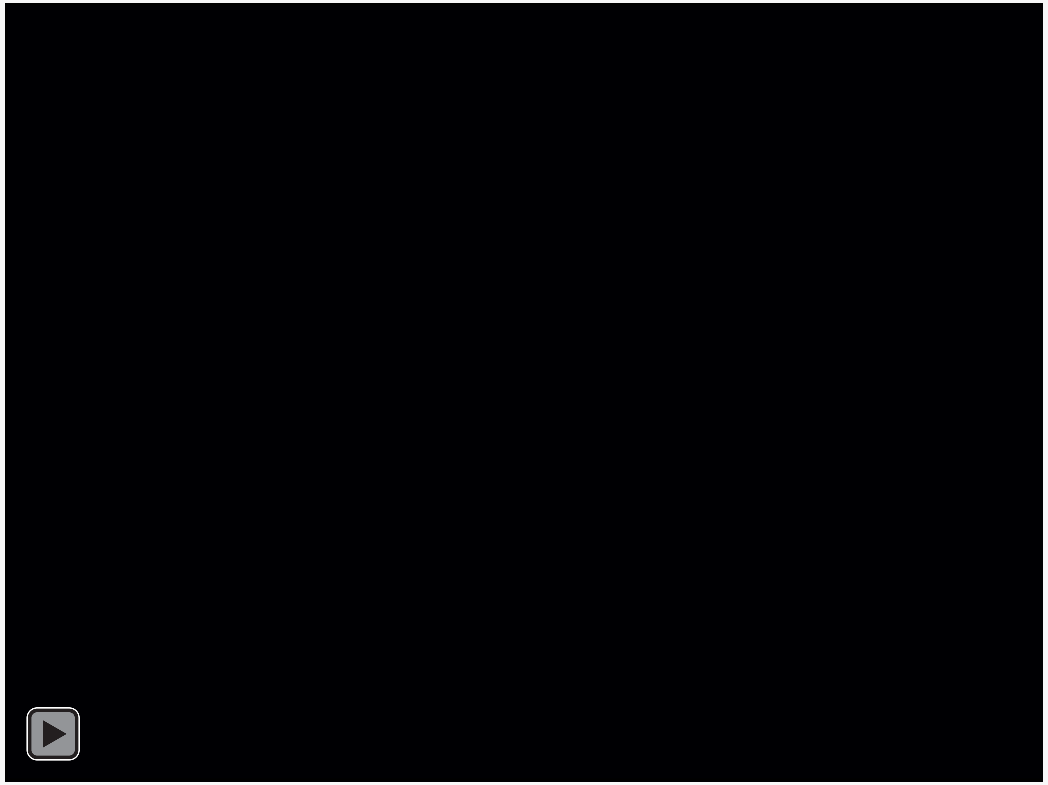
Switch is off



Switch is on

Results: Fluidic Switch

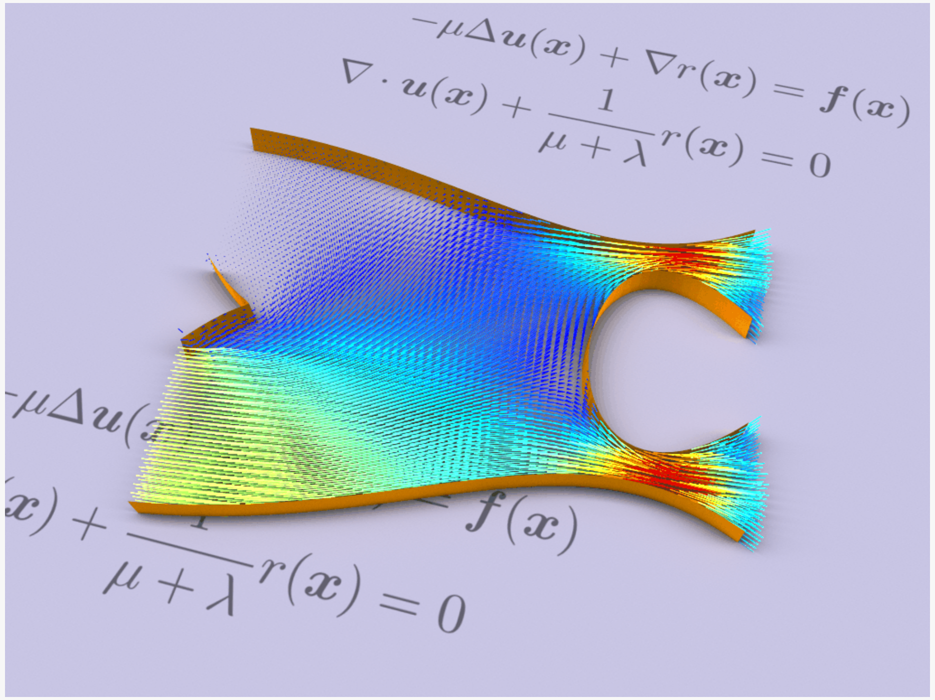
Optimization with multiple configurations

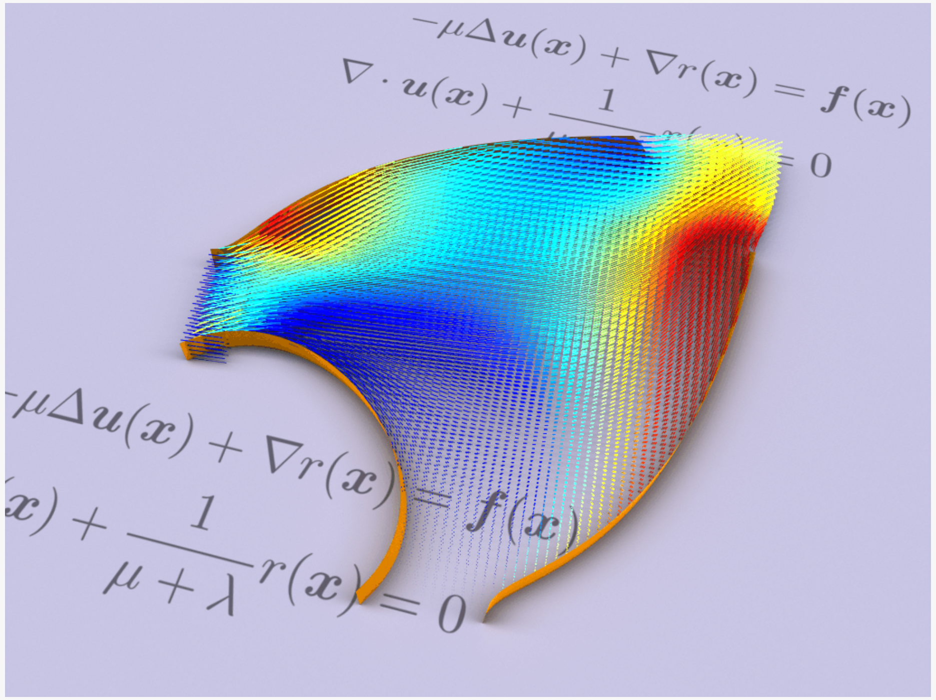


More Results

SIGGRAPH ASIA 2020 VIRTUAL

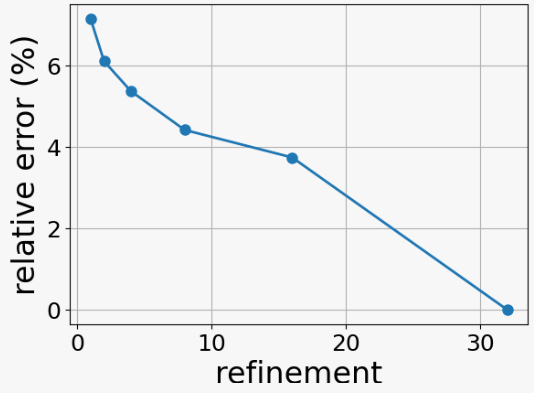
Fluid gates

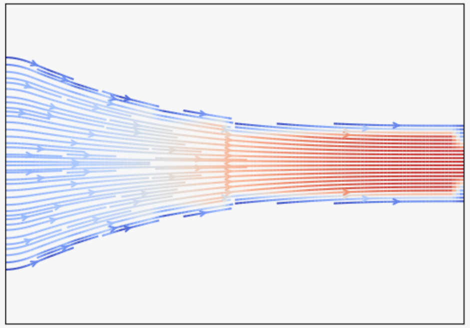


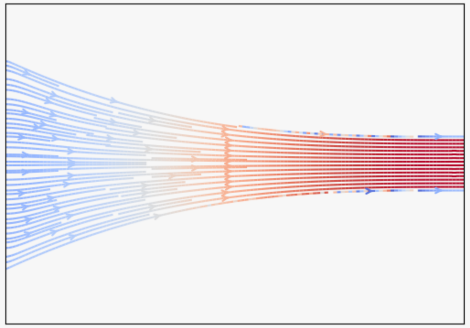


Results: Convergence Study

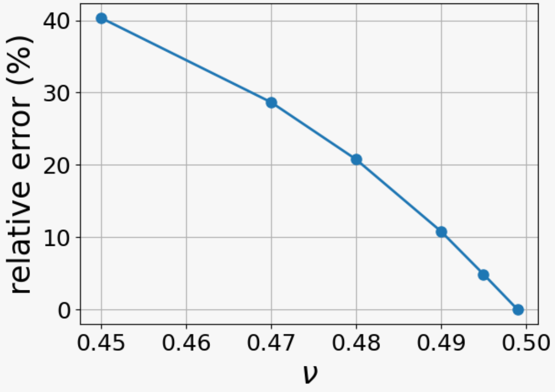
Simulating under refinement

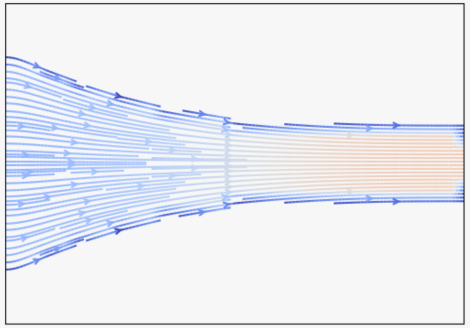


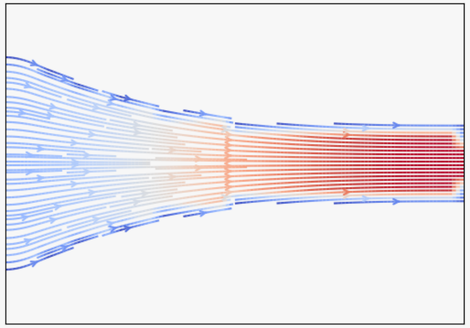

32 x 24


1024 x 768

Enforcing incompressibility

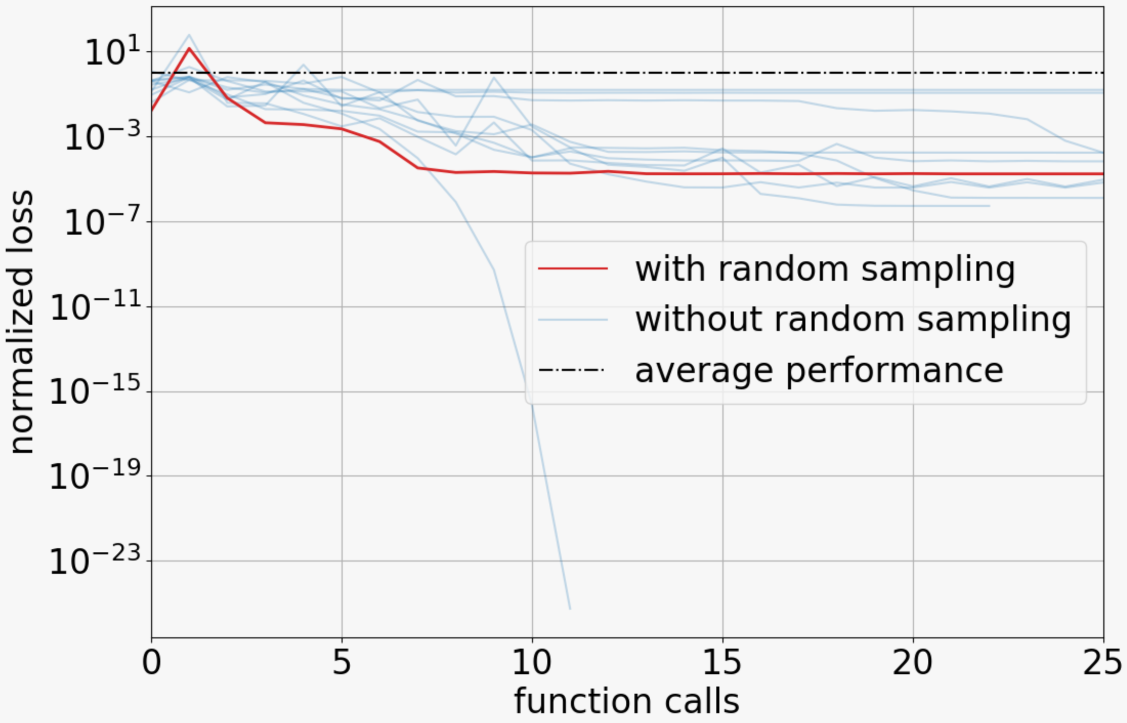


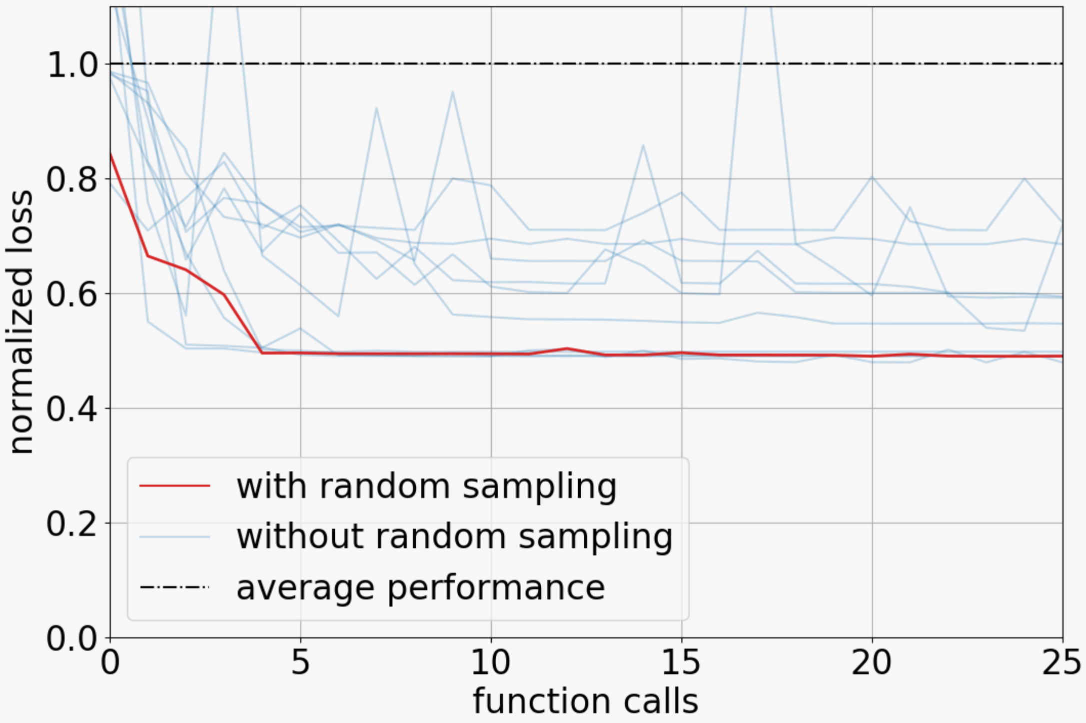

$\nu = 0.45$


$\nu = 0.499$

Ablation Study: Global Search

Comparisons between w/ and w/o sampling initial guesses





Summary

SIGGRAPH ASIA 2020 VIRTUAL

Differentiable simulation ⊃ applying the chain rule

Discretization and boundary conditions need careful treatment

Gradients speed up the process of finding optimal designs

…and they are more effective when combined with global search

Summary

Differentiable simulation $\supset$ applying the chain rule

Discretization and boundary conditions need careful treatment

Gradients speed up the process of finding optimal designs

…and they are more effective when combined with global search

Thank You for Watching

SIGGRAPH ASIA 2020 VIRTUAL

Code is available

GitHub link:

https://github.com/mit-gfx/diff_stokes_flow

or scan