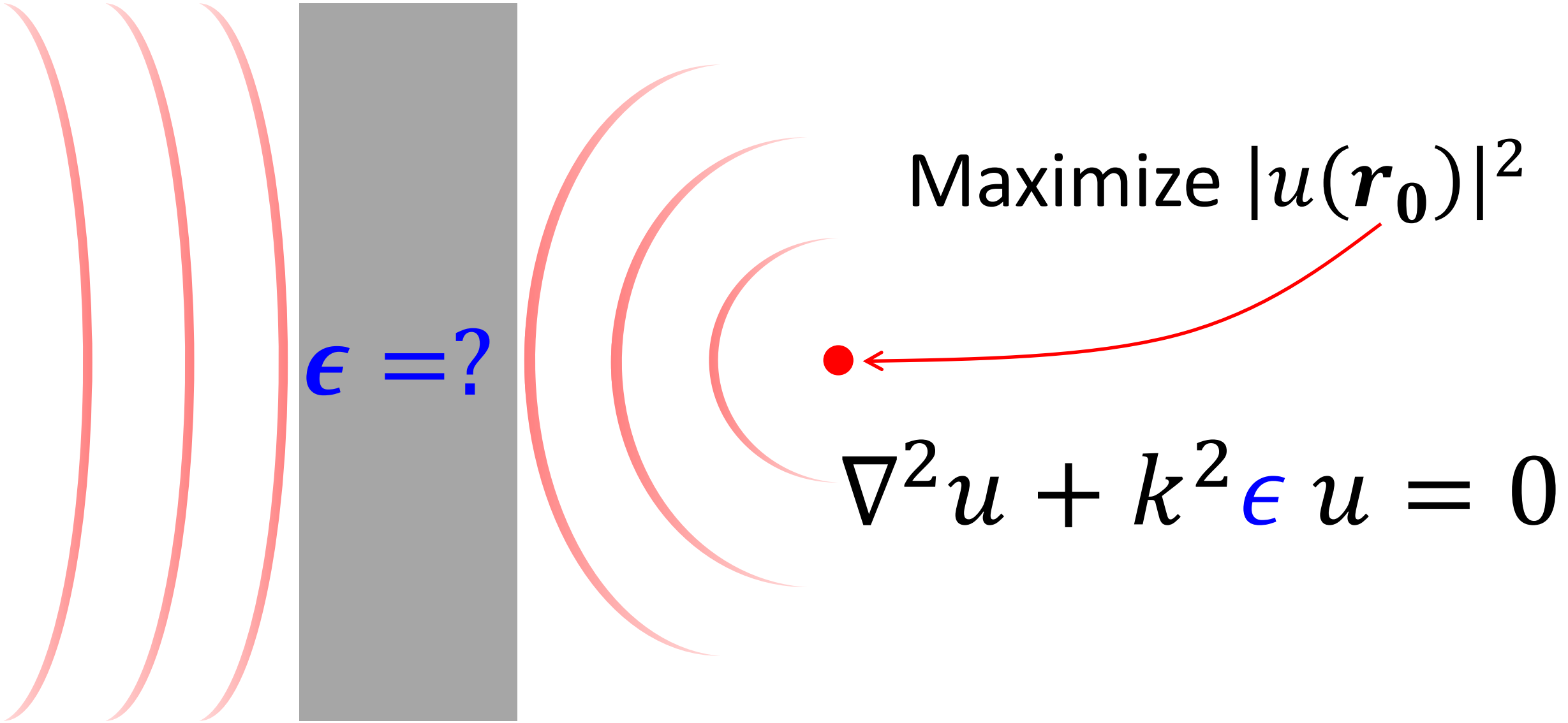
The adjoint method for differentiating complex computations ... with example applications in engineering design.

Dr. Zin Lin, MIT Applied Mathematics

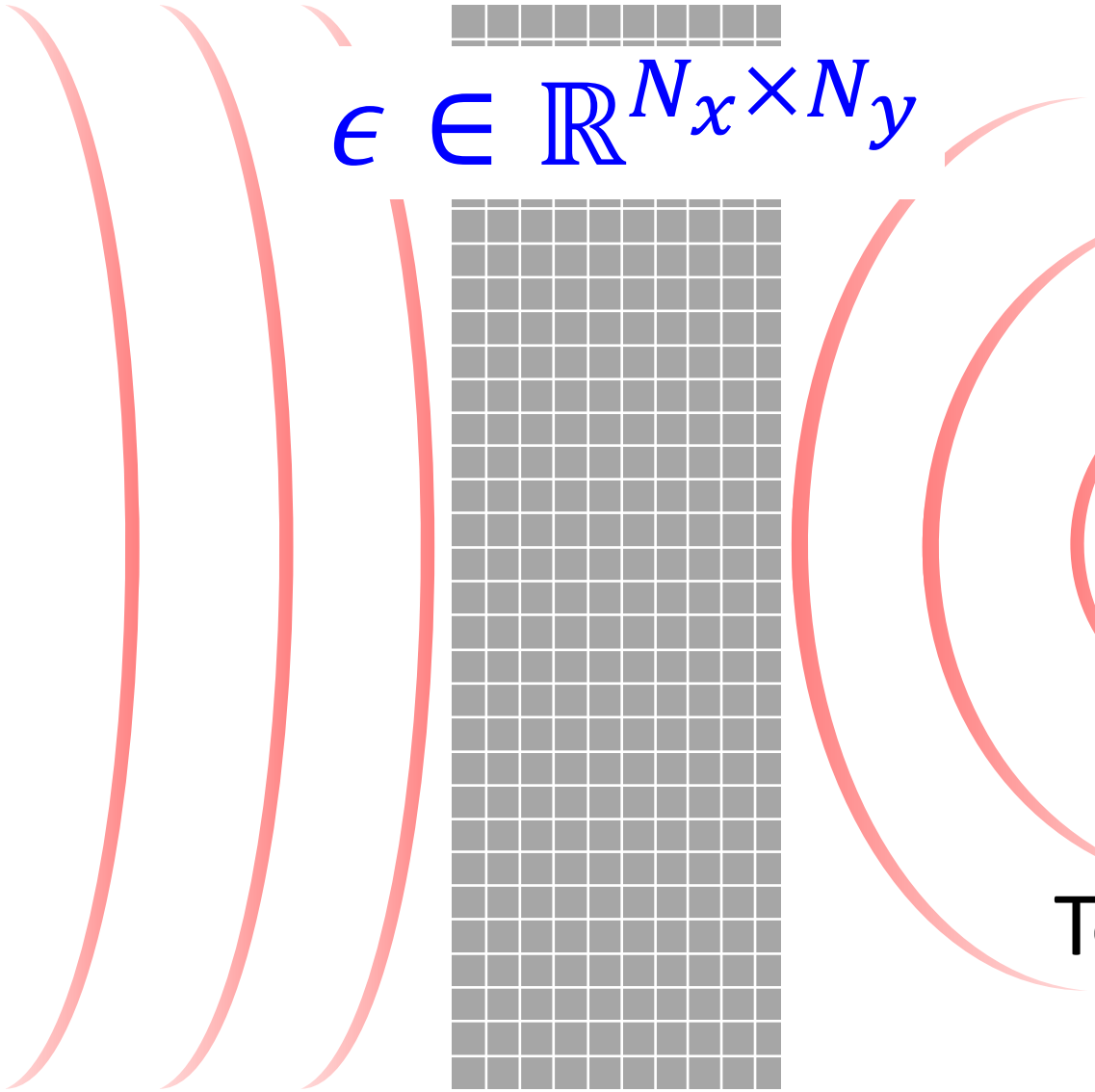
guest lecture for 18.335 at MIT, Spring 2021 with Prof. Steven Johnson

see also notes: https://math.mit.edu/~stevenj/18.336/adjoint.pdf

Example: optimizing *every* pixel of design…



Optimizing *every* pixel of your design ...



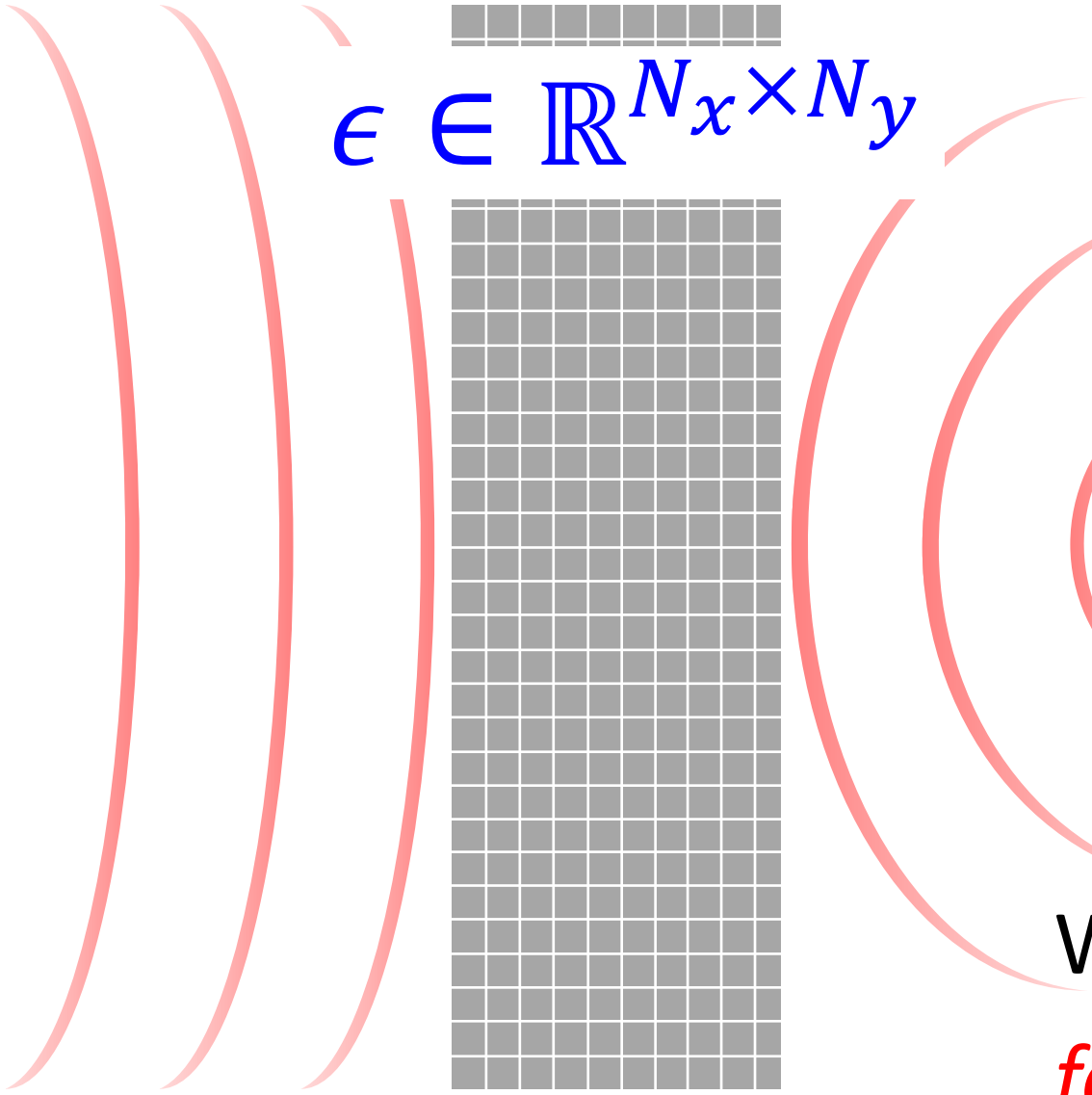
Maximize $|u(\boldsymbol{r_0})|^2$

$$\nabla^2 u + k^2 \epsilon\, u = 0$$

Total # of params. = $N_x \times N_y\ (\times N_z)$

$\sim 10^3 - 10^9$

Optimizing *every* pixel of your design …



Maximize $|u(\boldsymbol{r_0})|^2$

$$\nabla^2 u + k^2 \epsilon\, u = 0$$

We need to compute $\frac{\partial |u|^2}{\partial \epsilon} \in \mathbb{R}^{N_x \times N_y}$

fast and accurate.

Adjoint method (~ Back prop, Backward differentiation, etc)

Optimize $g(\boldsymbol{u}(\boldsymbol{p}), \boldsymbol{p})$

Subject to $\boldsymbol{A}(\boldsymbol{p})\boldsymbol{u} = \boldsymbol{b}(\boldsymbol{p})$

Requires $\frac{dg}{d\boldsymbol{p}} = ?$

gradient vector of size P

g is a scalar-valued function $\in \mathbb{R}$.

$\boldsymbol{u}$ is a "physics" vector of size N. (electric, displacement, pressure, temperature, etc.)

$\boldsymbol{p}$ is a param. vector of size P. (dielectric, mass density, conductivity, etc.)

$\boldsymbol{A}$ is an $N \times N$ matrix. (PDE: Maxwell, elastodynamics, heat equation, etc.)

Adjoint method (Linear equations)

$$\frac{dg}{d\boldsymbol{p}} = \underbrace{\frac{\partial g}{\partial \boldsymbol{p}}}_{1\times P} + \underbrace{\frac{\partial g}{\partial \boldsymbol{u}}}_{1\times N} \underbrace{\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{p}}}_{N\times P} \quad \text{(Chain rule)}$$

Adjoint method (Linear equations)

$$\frac{dg}{dp} = \frac{\partial g}{\partial p} + \frac{\partial g}{\partial u}\frac{\partial u}{\partial p}$$ (Chain rule)

$$\boldsymbol{Au = b} \quad \Rightarrow \quad \boldsymbol{A}\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{p}} = \frac{\partial \boldsymbol{b}}{\partial \boldsymbol{p}} - \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{p}}\boldsymbol{u}$$

Adjoint method (Linear equations)

$$\frac{dg}{d\boldsymbol{p}} = \frac{\partial g}{\partial \boldsymbol{p}} + \frac{\partial g}{\partial \boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{p}}$$ (Chain rule)

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b} \quad \Rightarrow \quad \boldsymbol{A} \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{p}} = \frac{\partial \boldsymbol{b}}{\partial \boldsymbol{p}} - \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{p}} \boldsymbol{u}$$

$$\frac{dg}{d\boldsymbol{p}} = \frac{\partial g}{\partial \boldsymbol{p}} + \frac{\partial g}{\partial \boldsymbol{u}} \left[\boldsymbol{A}^{-1} \left(\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{p}} - \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{p}} \boldsymbol{u} \right) \right]$$

Adjoint method (Linear equations)

$$\frac{dg}{d\boldsymbol{p}} = \frac{\partial g}{\partial \boldsymbol{p}} + \overbrace{\frac{\partial g}{\partial \boldsymbol{u}}}^{1\times N} \left[\overbrace{\boldsymbol{A}^{-1}}^{N\times N} \overbrace{\left(\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{p}} - \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{p}} \boldsymbol{u} \right)}^{N\times P} \right]$$

Adjoint method (Linear equations)

$$\frac{dg}{d\boldsymbol{p}} = \frac{\partial g}{\partial \boldsymbol{p}} + \overbrace{\frac{\partial g}{\partial \boldsymbol{u}}}^{1\times N} \underbrace{\left[\overbrace{\boldsymbol{A}^{-1}}^{N\times N} \overbrace{\left(\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{p}} - \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{p}} \boldsymbol{u} \right)}^{N\times P} \right]}_{N\times P}$$

☞ You have to solve $\boldsymbol{A}[\dots]_i = \frac{\partial \boldsymbol{b}}{\partial p_i} - \frac{\partial \boldsymbol{A}}{\partial p_i}\boldsymbol{u}$ for *every* $p_i,\ i = 1, \dots, P$!!!

Adjoint method (Linear equations)

$$\frac{dg}{dp} = \frac{\partial g}{\partial p} + \underbrace{\left[\underbrace{\frac{\partial g}{\partial u}}_{1\times N} \underbrace{\boldsymbol{A}^{-1}}_{N\times N}\right]}_{1\times N} \underbrace{\left(\frac{\partial \boldsymbol{b}}{\partial p} - \frac{\partial \boldsymbol{A}}{\partial p}\,\boldsymbol{u}\right)}_{N\times P}$$

****

KEY Trick:

switch the bracket

****

Adjoint method (Linear equations)

$$\frac{dg}{dp} = \frac{\partial g}{\partial p} + \underbrace{\left[\underbrace{\frac{\partial g}{\partial u}}_{1\times N}\ \underbrace{\boldsymbol{A}^{-1}}_{N\times N}\right]}_{1\times N}\underbrace{\left(\frac{\partial \boldsymbol{b}}{\partial p} - \frac{\partial \boldsymbol{A}}{\partial p}\,\boldsymbol{u}\right)}_{N\times P}$$

$$\boldsymbol{\lambda}^T = \frac{\partial g}{\partial \boldsymbol{u}}\boldsymbol{A}^{-1} \quad \Rightarrow \quad \boldsymbol{A}^T\boldsymbol{\lambda} = \frac{\partial g}{\partial \boldsymbol{u}}^T$$

Adjoint method (Linear equations)

$$\frac{dg}{d\boldsymbol{p}} = \frac{\partial g}{\partial \boldsymbol{p}} + \underbrace{\boldsymbol{\lambda}^T}_{1\times N} \overbrace{\left(\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{p}} - \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{p}}\,\boldsymbol{u}\right)}^{N\times P}$$

$$\boldsymbol{\lambda}^T = \frac{\partial g}{\partial \boldsymbol{u}}\boldsymbol{A}^{-1} \quad \Rightarrow \quad \boldsymbol{A}^T\boldsymbol{\lambda} = \frac{\partial g}{\partial \boldsymbol{u}}^T$$

Adjoint method (Linear equations)

Optimize $g(\boldsymbol{u}(\boldsymbol{p}), \boldsymbol{p})$

Subject to $\boldsymbol{A}(\boldsymbol{p})\boldsymbol{u} = \boldsymbol{b}(\boldsymbol{p})$

Requires $$\frac{dg}{d\boldsymbol{p}} = \frac{\partial g}{\partial \boldsymbol{p}} + \boldsymbol{\lambda}^T \left(\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{p}} - \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{p}} \boldsymbol{u} \right)$$

Where $$\boldsymbol{A}^T \boldsymbol{\lambda} = \frac{\partial g}{\partial \boldsymbol{u}}^T$$

Takeaway: you can obtain the entire gradient by solving the linear equations just one more time.

Adjoint method (Nonlinear equations)

Optimize $g(\boldsymbol{u}(\boldsymbol{p}), \boldsymbol{p})$

Subject to $\boldsymbol{f}(\boldsymbol{u}, \boldsymbol{p}) = 0$

Requires $\frac{dg}{d\boldsymbol{p}} = \frac{\partial g}{\partial \boldsymbol{p}} + \frac{\partial g}{\partial \boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{p}}$

$\boldsymbol{f}$ is a vector-valued function of size N. (nonlinear Maxwell equations, Navier-Stokes equations, etc.)

Adjoint method (Nonlinear equations)

$$\boldsymbol{f}(\boldsymbol{u},\boldsymbol{p}) = 0 \quad \Rightarrow \quad \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{p}} = -\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)^{-1} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}$$

$N \times N$ $\quad$ $N \times P$

$$\frac{d\boldsymbol{g}}{d\boldsymbol{p}} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}}\left[-\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)^{-1} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}\right]$$

Adjoint method (Nonlinear equations)

Again, switch the bracket

$$\frac{dg}{d\boldsymbol{p}} = \frac{\partial g}{\partial \boldsymbol{p}} - \left[\frac{\partial g}{\partial \boldsymbol{u}}\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)^{-1}\right]\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}$$

$$= \frac{\partial g}{\partial \boldsymbol{p}} - \boldsymbol{\lambda}^T \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}, \qquad \underbrace{\frac{\partial \boldsymbol{f}^T}{\partial \boldsymbol{u}}}_{N\times N}\boldsymbol{\lambda} = \frac{\partial g^T}{\partial \boldsymbol{u}}$$

Note: the adjoint problem is a linear problem (of lesser complexity than the forward nonlinear problem).

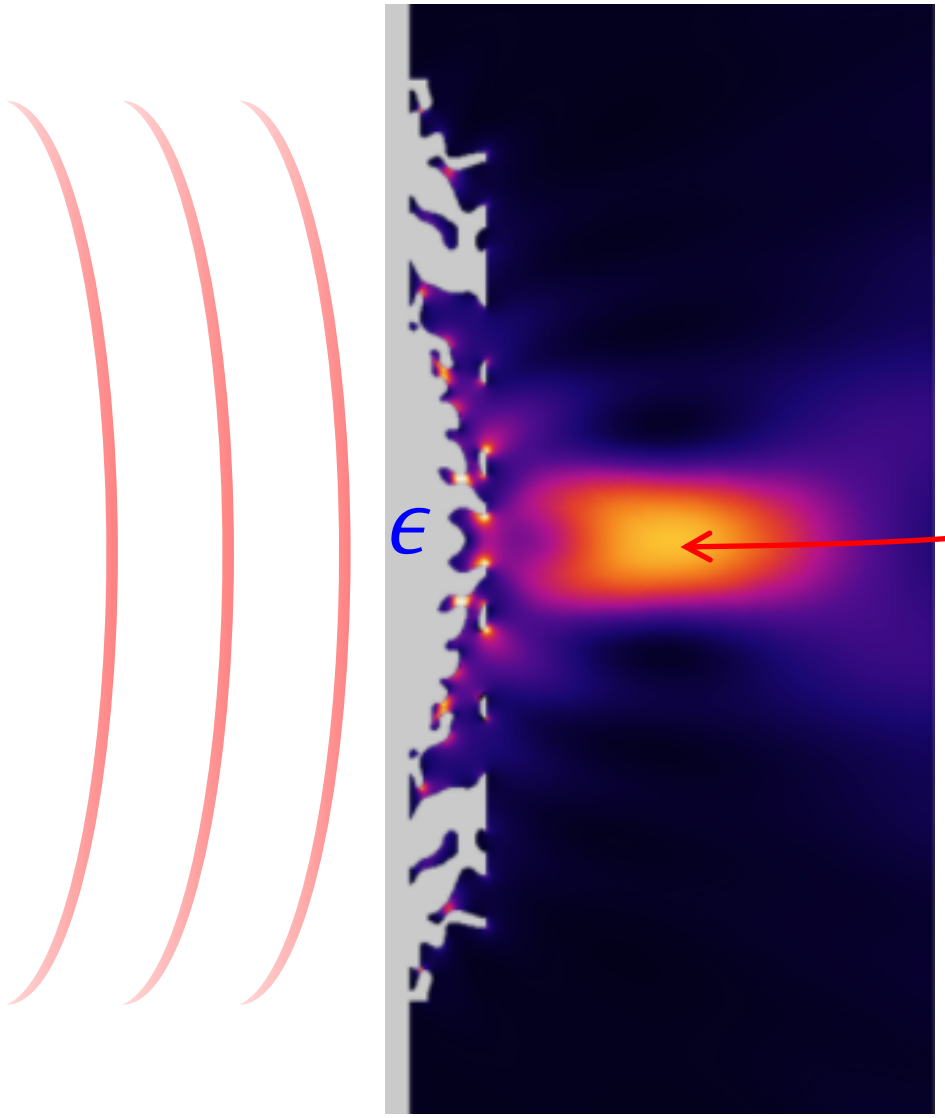
Example applications

Engineering optimization (structural "topology" optimization):

Find the physical structure that optimizes some objective (e.g. focusing light, minimizing drag, supporting weight, …).

Adjoint methods are key! "Forward" model is a huge simulation, but gradient requires only *one* additional simulation.

Topology-optimized nanophotonic lens

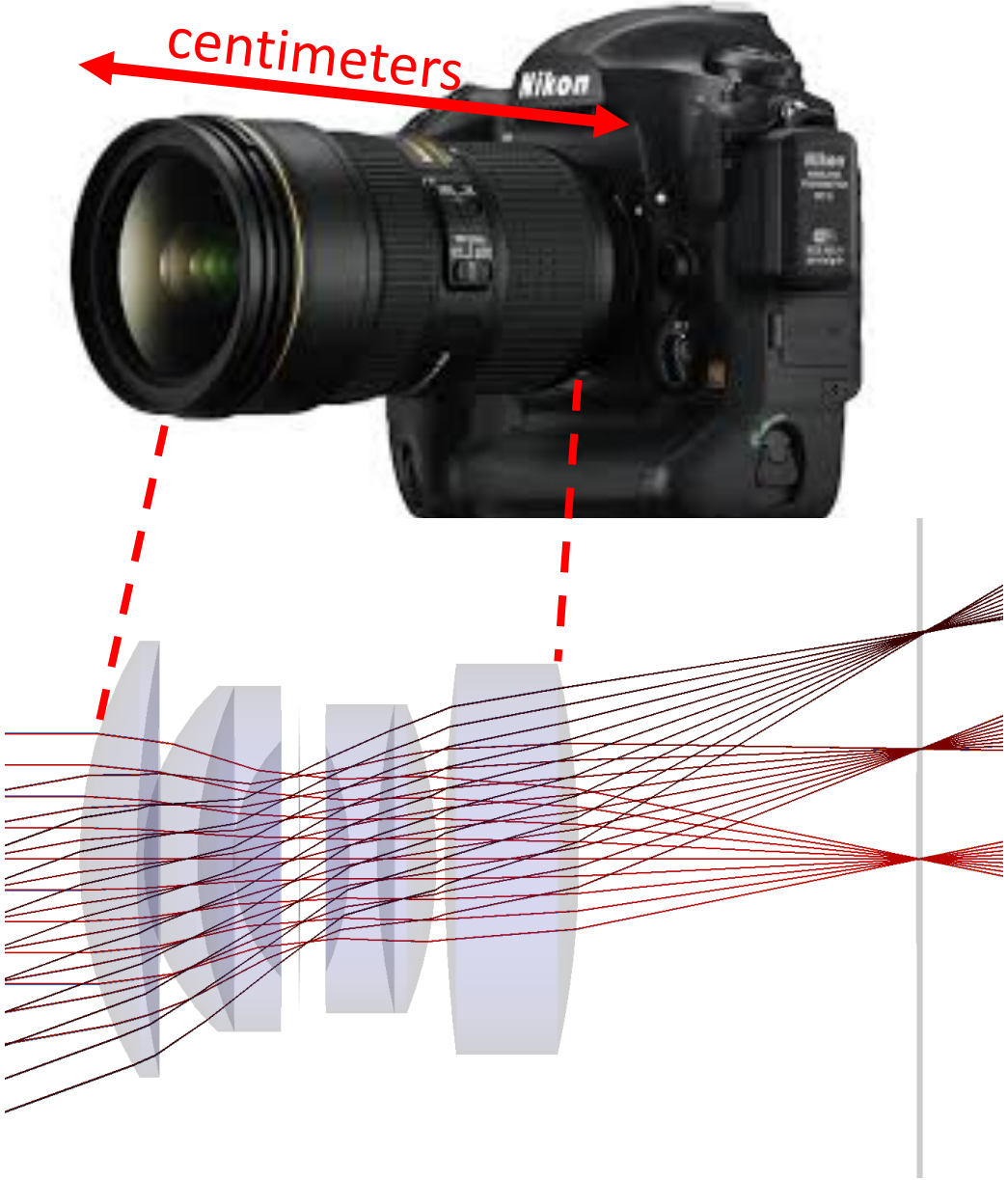


$$\text{Max. } |u(\boldsymbol{r_0})|^2$$

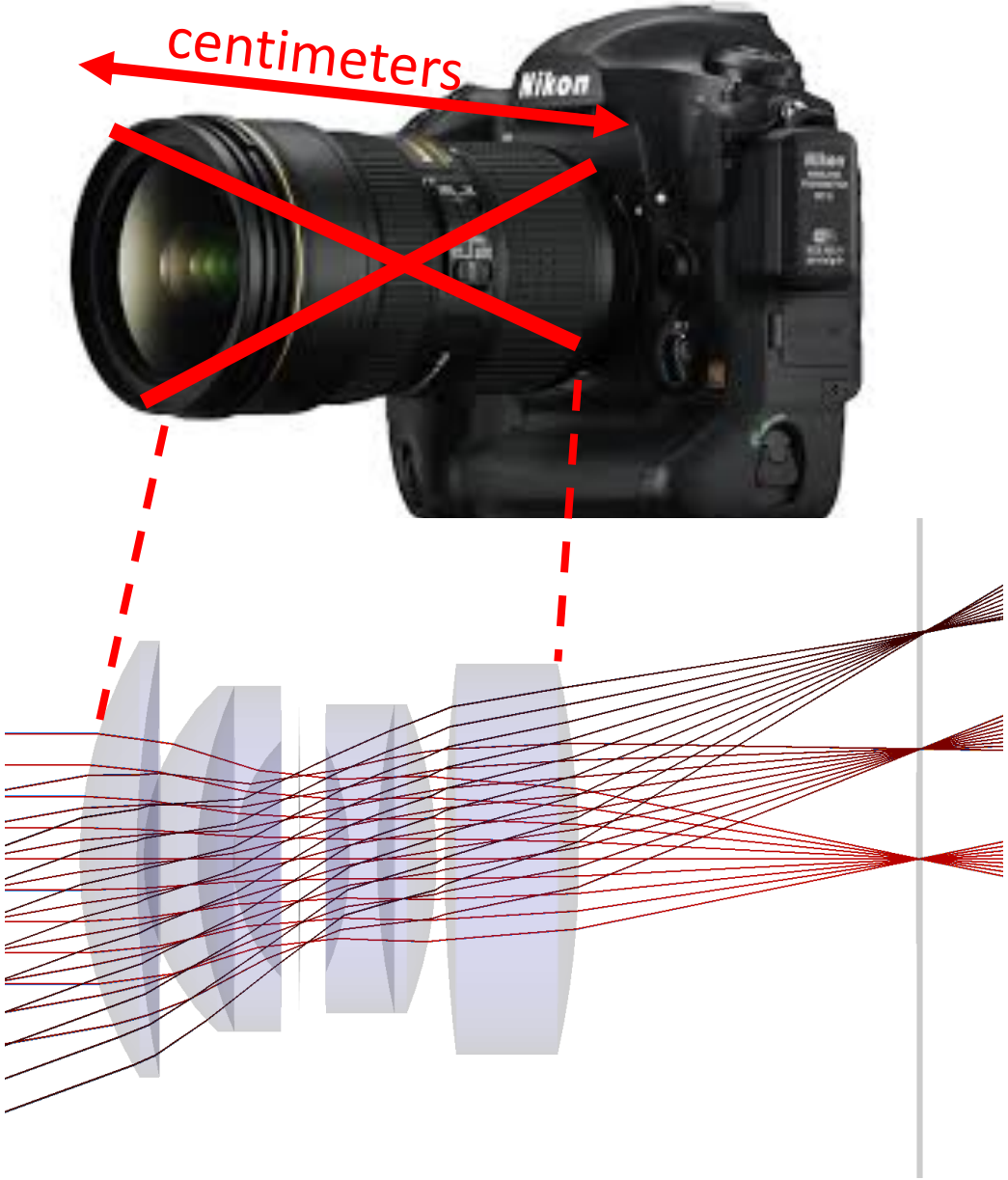
$$\nabla^2 u + k^2 \epsilon\, u = 0$$

Christiansen, Rasmus E., and Ole Sigmund. "Inverse design in photonics by topology optimization: tutorial." *JOSA B* 38.2 (2021): 496-509.

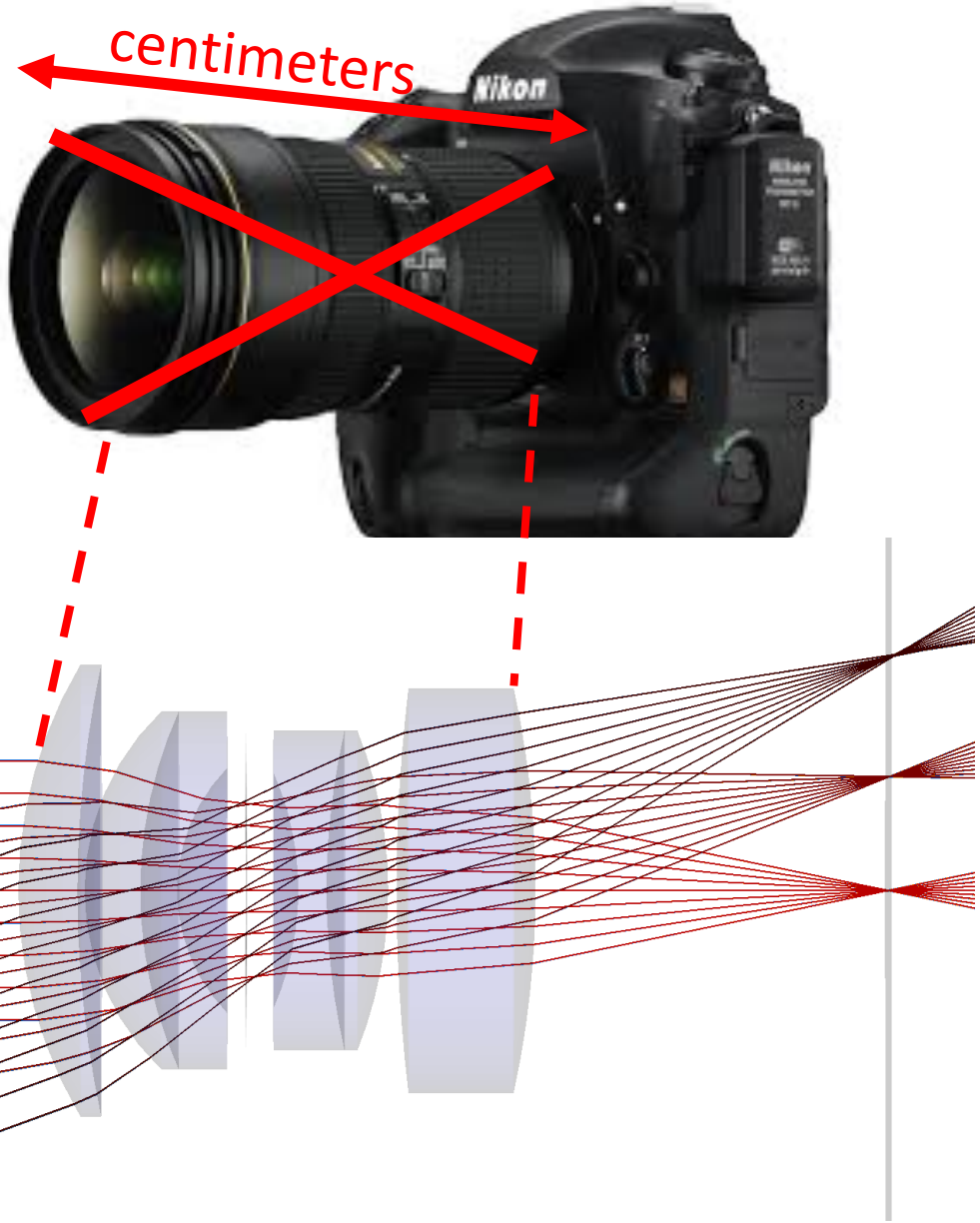
A squished camera lens?

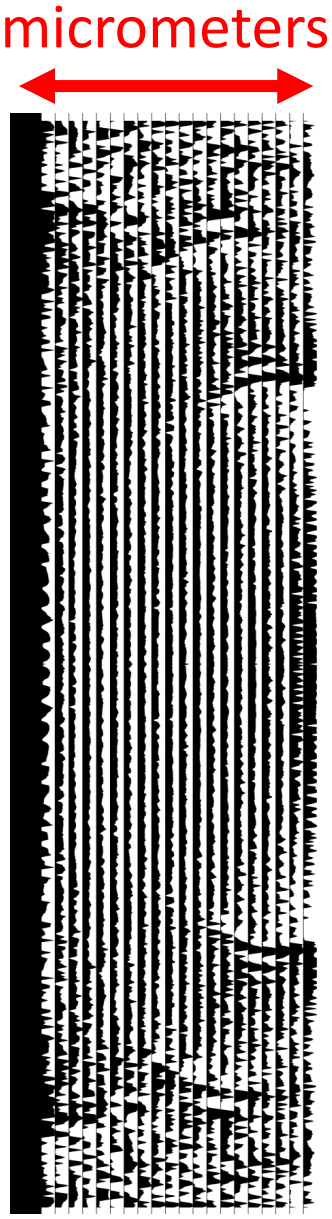


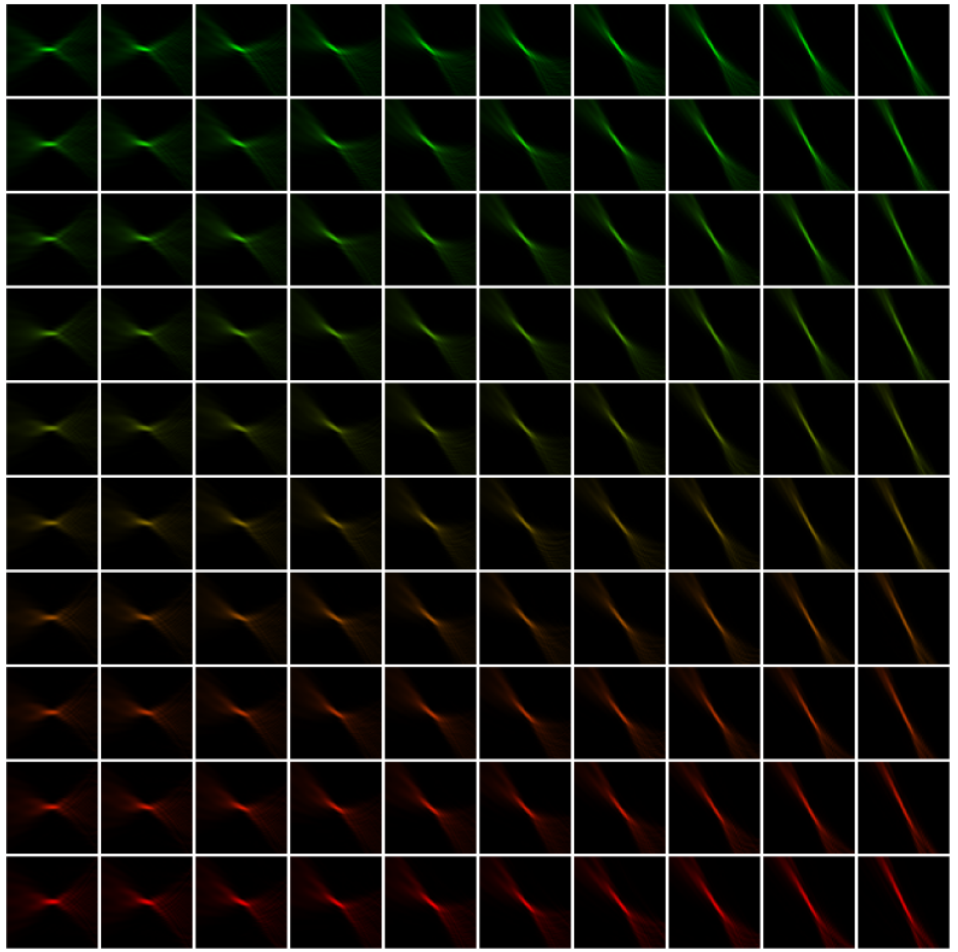
A squished camera lens?



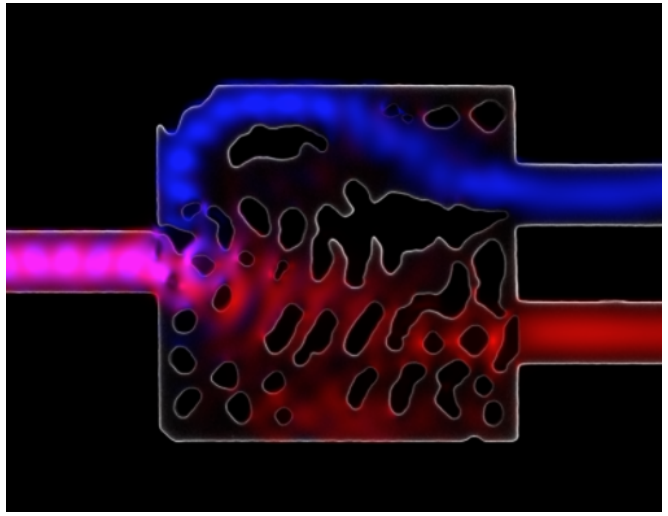
A squished camera lens?







Many more applications in optics and photonics …



Splitting wavelengths into different channels
Piggott et al, Nature Photonics (2016)

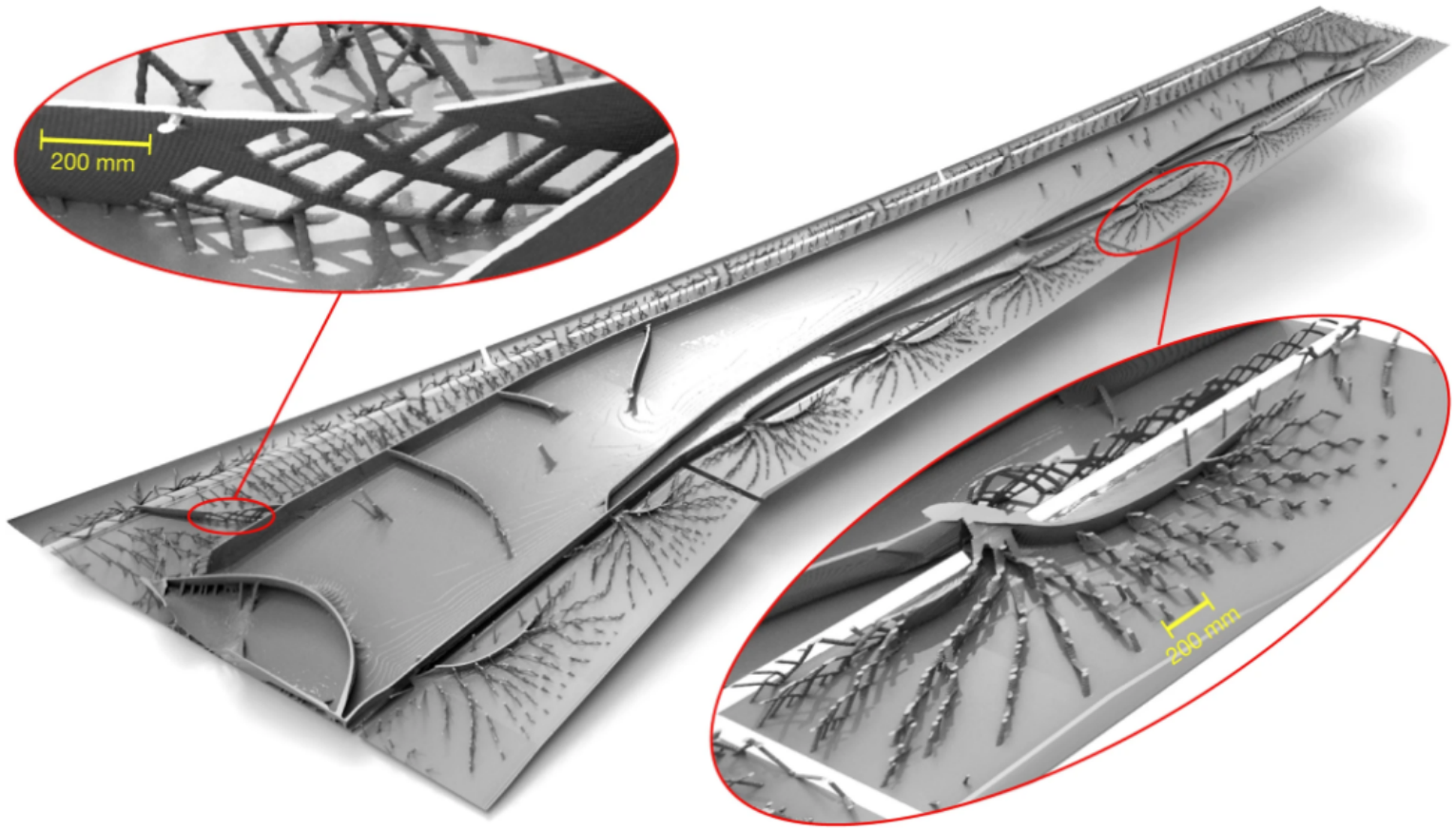
Trapping light in a very small volume
Wang et al, APL (2018)

Optimize $g(E(\boldsymbol{\epsilon}), \boldsymbol{\epsilon})$ over $\boldsymbol{\epsilon}$

Subject to $\nabla \times \nabla \times \boldsymbol{E} - k^2 \boldsymbol{\epsilon}\, \boldsymbol{E} = i\, \omega\, \boldsymbol{J}$

Topology-optimized aircraft wing

$\sim 10^9$ parameters



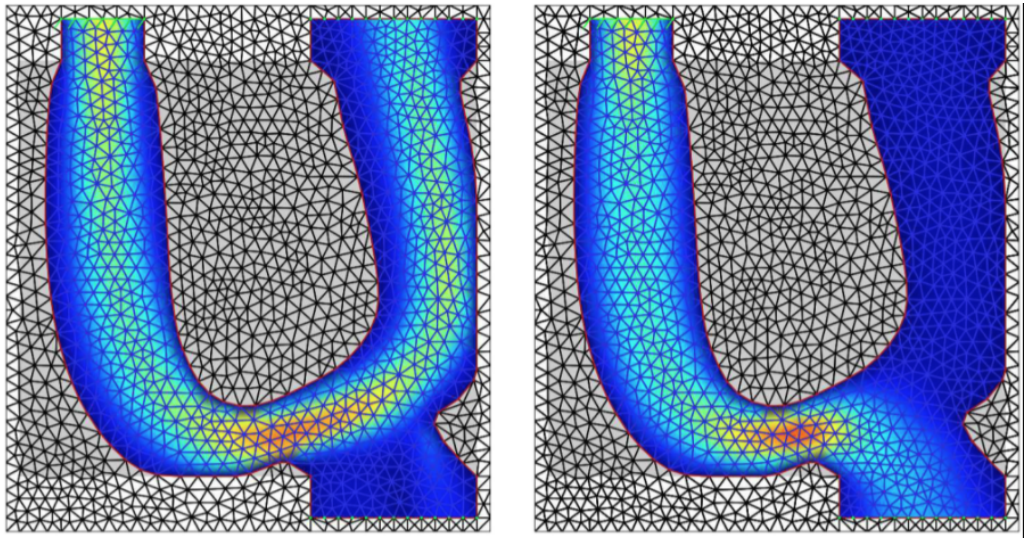
Goal: maximize stiffness under external loads, utilizing limited amount of material

→ Light but strong

→ 100s tonnes of fuel saving

Aage, Niels, et al. "Giga-voxel computational morphogenesis for structural design." *Nature* 550.7674 (2017): 84-86.

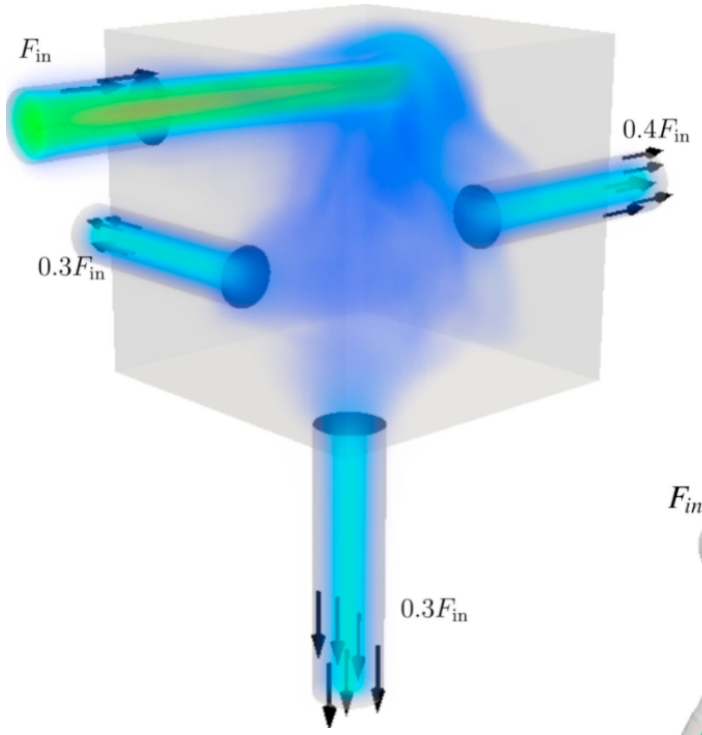
Topology optimization with fluid dynamics



(A) High *Re* flow, velocity magnitude (B) Low *Re* flow, velocity magnitude

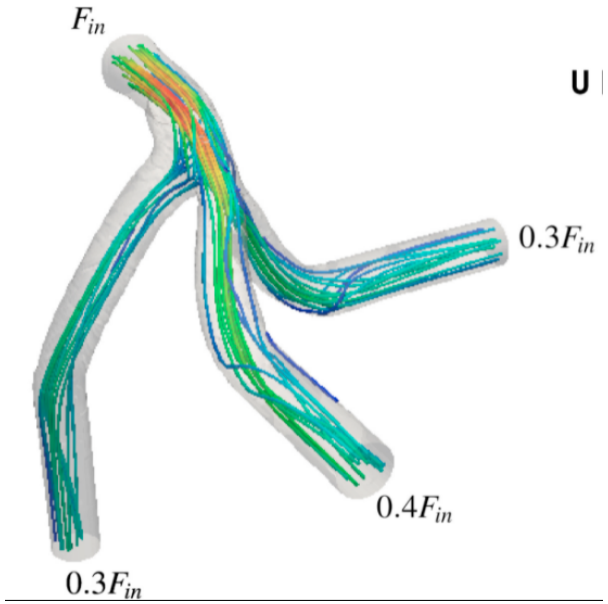
Switching flow channels for high vs. low viscosity

Zhou, Mingdong, et al. "Shape morphing and topology optimization of fluid channels by explicit boundary tracking." *International Journal for Numerical Methods in Fluids* 88.6 (2018): 296-313.



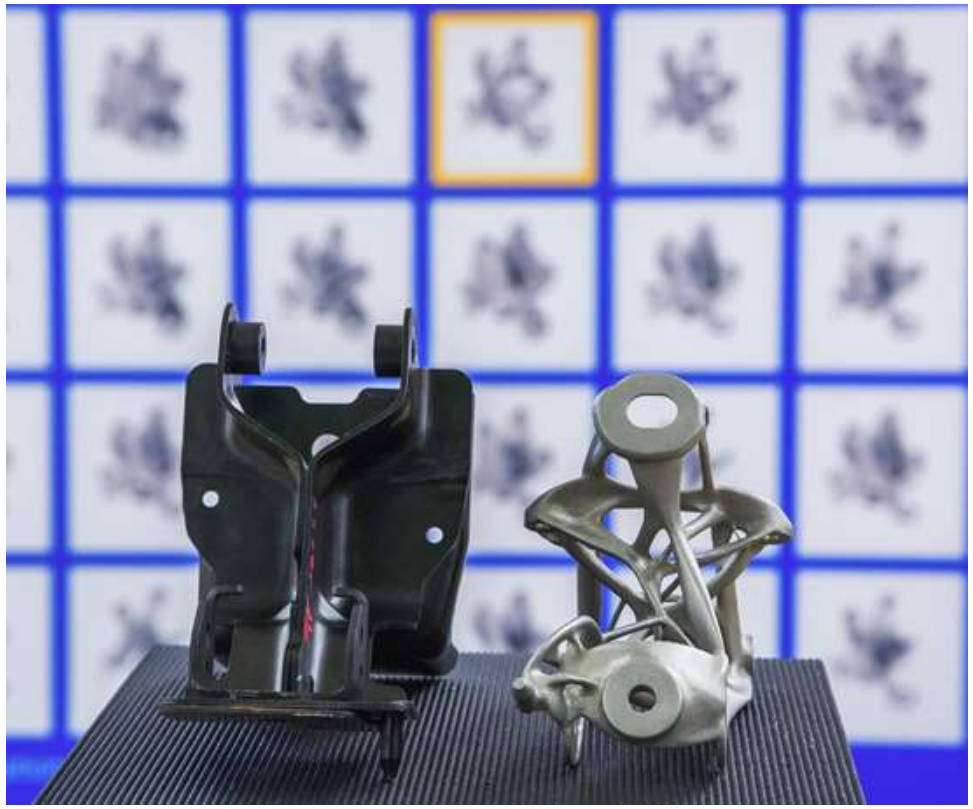
Dilgen, Cetin B., et al. "Topology optimization of turbulent flows." *Computer Methods in Applied Mechanics and Engineering* 331 (2018): 363-393.

Minimize the power dissipation due to turbulent flow for a given set of inlets and outlets.



Some industrial applications

Topology-optimized 3D-printed hip replacement (Altair)



Topology-optimized 3D-printed seat bracket (General Motors)

And many others …

Not just for physical problems!

Key point is that if you have *any* complicated calculation with lots of parameters, you can compute gradient (sensitivity) of a scalar output g(u) with respect to every parameter with roughly *one* additional calculation.

Enabling factor for large-scale optimization in machine learning [g = loss function, u = network outputs, **p** = network weights & other parameters], statistics, finance, and many other fields.

Central trick is to evaluate chain rule from "left-to-right", i.e. "back-propagating" in "reverse" order from "outputs to inputs", i.e. using left-to-right "vector–Jacobian products".