Automated design optimization via adjoint sensitivity analysis

This tutorial demonstrates the use of meep's support for *adjoint sensitivity analysis* to facilitate automated design optimization with derivative-based numerical optimizers. The python routines implementing this functionality are defined in `adjoint.py`.

Table of Contents

Preliminaries: Objective regions and design regions

The methods we'll illustrate are applicable to a broad class of optimization problems in which we seek to optimize Poynting fluxes and/or mode-expansion coefficients in one or more regions of a geometry by tuning the structure—that is, the spatial permittivity distribution $\epsilon(\mathbf{x})$—of some portion of the geometry.

More specifically, we'll consider problems with the following characteristics:

- The objective function is of the form

  $$F\Big(\{S_n\}, \{|\alpha_{nm}|^2\}\Big)$$

  where

  - $\{S_n\} = \{S_1, \cdots, S_N\}$ are the Poynting fluxes at some user-specified set of N flux regions $\{\mathcal{V}_1^o, \cdots, \mathcal{V}_N^o\}$, which we call the *objective regions* for the optimization problem

- α_{nm} is the eigenmode-expansion coefficient for the mth eigenmode in the nth objective region
- F is an arbitrary user-specified function of its inputs

- The quantity to be optimized is the permittivity distribution $\epsilon(\mathbf{x})$ within some user-specified volume $\mathcal{V}^d$, which we call the *design region*.
- The permittivity in the design region is expressed as an expansion in some (arbitrary, user-specified) set of basis functions:

$$\epsilon(\mathbf{x}) \equiv \sum_{p=1}^{P} a_p \psi_p(\mathbf{x}), \qquad \mathbf{x} \in \mathcal{V}^d$$

Here $\{\psi_p(\mathbf{x})\}, p = 1, \cdots, P$ is a set of P scalar-valued basis functions, defined for $\mathbf{x} \in \mathcal{V}^d$ the objective region, and the expansion coefficients $\{a_p\}$ are the variable parameters in our optimization.

Thus, problems for adjoint-based solvers in meep, you will specify

- a list of objective regions $\{\mathcal{V}^o_n\}$
- a design region $\mathcal{V}^d$
- a basis set $\{b_d(\mathbf{x})\}$
- an objective function F

Sample geometries

Basis functions and parameterized geometries

Specifying basis sets

The basis of expansion functions for the permittivity in the design region is described by a single routine that inputs the normalized coordinates of a point $\mathbf{x}$ in the design region and returns a vector of length P containing $[\psi_1(\mathbf{x}), \cdots, \psi_P(\mathbf{x})]$. Here "normalized coordinates" refers to a shifted and scaled coordinate system in which the design region is centered at the origin and has length 1 in all directions; so that the normalized coordinates $\overline{\mathbf{x}}$ for points in the design region satisfy $-0.5 < \overline{x}_i < 0.5$ for all i. then every component of $\overline{\mathbf{x}}$ lies in the range $[-0.5, 0.5]$

Here are some examples of basis sets for 2D design regions:

- A simple polynomial basis set with 4 functions $\{1, x, y, xy\}$:

```
def basis(pbar):
    x=pbar[0]
    y=pbar[1]
    return [1,x,y,x*y]
```

- A basis set for the circular-hole example above that contains a single function that vanishes outside a circle of (normalized) radius 0.25 and is 1 inside that circle:

```
def basis(pbar):
    x=pbar[0]
    y=pbar[1]
    return 1 if (x*x+y*y)<=0.25*0.25 else 0.0
```

(Note that normalized radius 0.5 would correspond to the maximal inscribed circle in the design region.)

The file `adjoint.py` defines some convenient predefined basis sets:

- `fourier_basis(kxMax, kyMax)`

  2D Fourier basis. `kxMax, kyMax` are integers specifying the maximum spatial frequency in each direction as a multiple of the base frequency. The size of the basis is `(2*kxMax+1)*(2*kyMax+1).` (The return values of e.g. `fourier_basis(4,3)` is a function that may be passed as the `basis` parameter to functions like `custom_dielectric` below.)
- `annular_basis(NR=2, kMax=2, rho_max=0.5, rho_min=0.0)`

  A basis for a disc or annular region, consisting of a sinusoids in the angular direction paired with Legendre polynomials in the radial direction.

Parameterized geometries

Having defined a set of basis functions, a design region with parametrized permittivity described by a basis-set function `basis` and a vector of basis-coefficients `coeffs` may be added to a meep geometry by including the following block among its objects:

```
mp.Block(center=design_center, size=design_size,
                        material=custom_dielectric(design_center, design_size, basis, coeffs)
        )
```

Here `design_center, design_size` are the center and size of the design region and `custom_dielectric` is a function defined in `adjoint.py` that defines a meep material function given by the weighted basis set.

Computing objective functions and gradients

Having defined an optimization problem, we compute the value of the objective function, and its gradient with respect to the expansion coefficients, by calling

```
f,gradf = get_objective_and_gradient(sim, forward_sources,
                                     objective_regions, design_region, basis,
                                     objective)
```

where the inputs are

- `sim` : `simulation` containing your geometry (but no sources, DFT regions, etc.)
- `forward_sources` : sources for the fields considered by your objective function.
- `objective_regions` : List of DFT regions on which arguments to your objective function are defined.
- `design_region` : Design region of your problem
- `basis` : Function describing basis set, as defined above
- `objective` : Function describing objective function, as defined above