Nanophotonic Inverse Design

Jesse Lu

December 13, 2011

Contents

1 Introduction

Our goal is to produce a software package capable of designing virtually any nanophotonic device. The software must produce designs

- with exceptional device performance,
- which are easily manufacturable, and
- with computational efficiency.

Such a software package would be extremely useful in designing the components needed to guide light on a computer chip such as couplers, filters, absorbers, and multiplexors.

However, the scale of the problem as well as the difficulty of inverting the underlying wave equation have been major obstacles in producing successful design algorithms.

1.1 Problem statement

We formulate the design problem in the following way,

$$\begin{aligned} \text{minimize} \quad & f(x) + g(p) && \text{(1a)} \\ \text{subject to} \quad & r(x, p) = 0 && \text{(1b)} \end{aligned}$$

where

- x is the field variable,
- p is the structure variable,
- $f(x)$ is the performance objective,
- $g(p)$ is the manufacturability objective,
- $f(x) + g(p)$ is generally referred to as the design objective, and
- $r(x, p)$ is the physics residual for which we use the time-harmonic electromagnetic wave equation,

$$r(x, p) = (\nabla \times \epsilon^{-1} \nabla \times -\mu\omega^2)H - \text{sources}. \tag{2}$$

1.1.1 Multimode formulation

We often desire a single device to perform multiple functions, in this case we can modify (1) in the following way,

$$\begin{aligned} \text{minimize} \quad & \sum_i f_i(x_i) + g(p) && \text{(3a)} \\ \text{subject to} \quad & r_i(x_i, p) = 0. && \text{(3b)} \end{aligned}$$

1.2 Key insights

Generic nonlinear optimization routines are usually unable to solve (1), because there are an extremely large (millions) of variables, and because $r(x,p)$ often results in ill-conditioned matrices. For this reason, we need to take advantage of key features of the problem.

- $r(x,p)$ is separably affine (bi-affine) in x and p,

$$r(x,p) = A(p)x - b(p) = B(x)p - d(x), \tag{4}$$

  this allows us to form two simpler sub-problems.

- Simulators which compute $A(p)^{-1}z$ are available, where z is an arbitrary vector, even for very large systems (millions of variables).

- Solving $B(x)p - d(x) = 0$ is possible with generic software, because manufacturing processes severely limit the degrees of freedom of p (decreasing p to thousands of variables).

2 Adjoint method

The adjoint method is a steepest-descent method on the space $r(x,p) = 0$, and relies upon the following linear approximations of the design objective and the physics residual,

$$(f(x+\Delta x) + g(p+\Delta p)) - (f(x) + g(p)) \approx \nabla f^T \Delta x + \nabla g^T \Delta p \tag{5a}$$

$$r(x+\Delta x, p+\Delta p) - r(x,p) \approx A(p)\Delta x + B(x)\Delta p. \tag{5b}$$

Assuming a starting point already satisfying $r(x,p) = 0$, we note that (5b) must equal zero, in order to keep the physics residual at zero. This allows us to form the following relationship between Δx and Δp,

$$A(p)\Delta x + B(x)\Delta p = A\Delta x + B\Delta p = 0 \tag{6a}$$

$$\Delta x = -A^{-1}B\Delta p, \tag{6b}$$

which allows us to write (5a) only in terms of Δp,

$$\nabla f^T \Delta x + \nabla g^T \Delta p = -(\nabla f^T A^{-1} B - \nabla g)\Delta p. \tag{7}$$

Thus, we see that the steepest-descent direction is

$$\Delta p = B^T A^{-T} \nabla f - \nabla g. \tag{8}$$

2.1 Computational cost

The adjoint method proceeds by iteratively

1. updating p along Δp, and then

2. updating x by solving $r(x, p)$ for fixed p.

Computationally, step 1 requires an A^{-T} solve, and step 2 requires an A^{-1} solve. The strength of the adjoint method resides in the fact that each of these operations corresponds to a *single simulation*, and so the entire iteration costs only two simulations.

2.2 Multi-mode formulation

If one wishes to design a multi-functional device, the adjoint method can also be applied to (3). In this case,

$$\Delta p = \sum_i B_i^T A_i^{-T} \nabla f_i - \nabla g, \tag{9}$$

and each iterations costs $2N$ simulations, where N is the number of modes considered. Note that the $2N$ simulations can be spread over N different computational nodes, so that the total running time of the optimization is virtually identical to the single-mode case.

3 Alternating directions method

An alternative optimization method is to iteratively solve

$$\underset{x}{\text{minimize}} \quad f(x) + \frac{\rho}{2}\|r(x,p)\|^2 \tag{10a}$$
$$\underset{p}{\text{minimize}} \quad g(p) + \frac{\rho}{2}\|r(x,p)\|^2, \tag{10b}$$

where the coefficient ρ can be gradually increased in order to drive $r(x, p) \to 0$.

The alternating directions method allows a non-physical starting point, and takes advantage of bi-affine $r(x, p)$ in the sense that the subproblems can be re-written as

$$\underset{x}{\text{minimize}} \quad f(x) + \frac{\rho}{2}\|A(p)x - b(p)\|^2 \tag{11a}$$
$$\underset{p}{\text{minimize}} \quad g(p) + \frac{\rho}{2}\|B(x)p - d(x)\|^2, \tag{11b}$$

since each of these subproblems may be readily solved by generic optimization tools, especially if both $f(x)$ and $g(p)$ are convex.

4 Alternating directions method of multipliers (ADMM)

The inclusion of a dual variable (y) allows us to form an *augmented Lagrangian*,

$$\mathcal{L}(x, p, y) = f(x) + g(p) + \frac{\rho}{2}\|r(x,p)\|^2 + y^T r(x,p). \tag{12}$$

This leads to the alternating directions method of multipliers (ADMM) [1], which proceeds in the following way,

$$x := \operatorname*{argmin}_x \mathcal{L}(x, p, y) \tag{13a}$$

$$p := \operatorname*{argmin}_p \mathcal{L}(x, p, y) \tag{13b}$$

$$y := y + \rho r(x, p). \tag{13c}$$

Similarily to alternating directions, ADMM allows for an infeasible starting point, $r(x, p) \neq 0$. However, ADMM allows the coefficient ρ to remain fixed, and generally demonstrates faster convergence.

4.1 Computational cost of ADMM

We infer that the majority of computational resources will be used in updating x since the y-update is trivial, and the p-update involves far fewer variables (see section 1.2).

In this vein, we examine the computational work, for various choices of $f(x)$, required in efficiently solving

$$\operatorname*{argmin}_x \mathcal{L}(x, p, y) = \operatorname*{argmin}_x f(x) + \frac{\rho}{2}\|Ax - b\|^2 + y^T(Ax - b) \tag{14}$$

via Newton's method; that is, updating x along Δx given by

$$\Delta x = (\nabla^2_{xx}\mathcal{L})^{-1}\nabla_x \mathcal{L}, \tag{15}$$

where

$$\nabla_x \mathcal{L} = \nabla f(x) + \rho A^T(Ax - b + \frac{1}{\rho}y) \tag{16a}$$

$$\nabla^2_{xx}\mathcal{L} = \nabla^2 f(x) + \rho A^T A. \tag{16b}$$

Furthermore, we limit our analysis to choices of $f(x)$ for which $\mathcal{L}(x, p, y)$ is quadratic, since the optimum is simply $x + \Delta x$ and requires only one calculation of Δx. Newton's method, of course, is very capable of minimizing nonlinear functions, and adapting our analysis to nonlinear choices of $f(x)$ simply requires multiple steps (Δx) to be computed.

4.1.1 Linear $f(x)$

We investigate the choice $f(x) = c^T x$, where $c \in \mathbf{R}^{n\times 1}$ and $x \in \mathbf{R}^{n\times 1}$. Here,

$$\nabla f(x) = c \tag{17a}$$

$$\nabla^2 f(x) = 0 \tag{17b}$$

which results in

$$\nabla_x \mathcal{L} = c + \rho A^T(Ax - b + \frac{1}{\rho}y) = \rho A^T(Ax - b + \frac{1}{\rho}(y + \tilde{c})) \tag{18a}$$

$$\nabla^2_{xx}\mathcal{L} = \rho A^T A, \tag{18b}$$

where $\tilde{c} = A^{-T}c$. A single Newton step is

$$\begin{aligned} \Delta x &= (\nabla^2_{xx}\mathcal{L})^{-1}\nabla_x\mathcal{L} \\ &= (\rho^{-1}A^{-1}A^{-T})A^T(Ax - b + \frac{1}{\rho}(y + \tilde{c})) \\ &= A^{-1}(Ax - b + \frac{1}{\rho}(y + \tilde{c})). \end{aligned} \tag{19}$$

From this analysis we see that the computation cost of a linear performance objective, $f(x)$, for the x-update is two simulations. That is, in a very similar fashion to the adjoint method, an A^{-T} solve is required to compute $\tilde{c}$, and an additional A^{-1} solve is needed to compute Δx.

4.1.2 Low-rank quadratic $f(x)$

We investigate the case where $f(x) = \frac{1}{2}\|C_1^T x - d_1\|^2$, where $C_1 \in \mathbf{R}^{n\times p_1}$ and $d_1 \in \mathbf{R}^{p_1\times 1}$, for $p_1 \ll n$. Here,

$$\nabla f(x) = C_1(C_1^T x - d_1) \tag{20a}$$

$$\nabla^2 f(x) = C_1 C_1^T \tag{20b}$$

which results in

$$\nabla_x\mathcal{L} = \rho A^T((I + \frac{1}{\rho}\tilde{C}_1\tilde{C}_1^T)Ax - b + \frac{1}{\rho}(y - \tilde{C}_1 d)) \tag{21a}$$

$$\nabla^2_{xx}\mathcal{L} = \tilde{C}_1\tilde{C}_1^T + \rho A^T A, \tag{21b}$$

where $\tilde{C}_1 = A^{-T}C_1$.

We then apply the matrix inversion lemma (see appendix C.2)

$$\begin{aligned} (\nabla^2_{xx}\mathcal{L})^{-1} &= (\tilde{C}_1\tilde{C}_1^T + \rho A^T A)^{-1} \\ &= \frac{1}{\rho}A^{-1}(I - \tilde{C}_1(\rho I + \tilde{C}_1^T\tilde{C}_1)^{-1}\tilde{C}_1^T)A^{-T} \\ &= \frac{1}{\rho}A^{-1}MA^{-T} \end{aligned} \tag{22}$$

where $M = I - \tilde{C}_1(\rho I + \tilde{C}_1^T\tilde{C}_1)^{-1}\tilde{C}_1^T$. Note that computing M is computationally inexpensive since the matrix which must be inverted, $\rho I + \tilde{C}_1^T\tilde{C}_1$, is $p_1 \times p_1$ ($p_1 \ll n$).

The Newton step is given by

$$\Delta x = A^{-1}M((I + \frac{1}{\rho}\tilde{C}_1\tilde{C}_1^T)Ax - b + \frac{1}{\rho}(y - \tilde{C}_1 d_1)), \tag{23}$$

and requires p_1 solutions of A^{-T} to compute $\tilde{C}_1$, and then 1 solution of A^{-1} to compute Δx.

4.1.3 $f(x)$ composed of linear equality constraints

We investigate the situation where $f(x)$ involves satisfying the equality constraint $C_2^T x = d_2$, where $C_2 \in \mathbf{R}^{n \times p_2}$ and $d_2 \in \mathbf{R}^{p_2 \times 1}$. Once again, we assume that $p_2 \ll n$.

Newton's method for a linearly-constrained quadratic function is

$$\begin{bmatrix} \nabla_{xx}^2 \mathcal{L} & C_2 \\ C_2^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ v \end{bmatrix} = \begin{bmatrix} -\nabla_x \mathcal{L} \\ C_2 x - d_2 \end{bmatrix}. \tag{24}$$

We assume that

$$(\nabla_{xx}^2 \mathcal{L})^{-1} = A^{-1} M A^{-T} \tag{25a}$$

$$\nabla_x \mathcal{L} = A^T z \tag{25b}$$

and solve for Δx using block elimination (see appendix C.3),

$$v = -(\tilde{C}_2^T M \tilde{C}_2)^{-1} (\tilde{C}_2^T A x - d_2 + \tilde{C}_2^T M z) \tag{26a}$$

$$\Delta x = -A^{-1} M (z + \tilde{C}_2 v), \tag{26b}$$

where $\tilde{C}_2 = A^{-T} C_2$. If we condense the block elimination procedure to a single step, we obtain

$$\Delta x = -A^{-1} M (z - \tilde{C}_2 (\tilde{C}_2^T M \tilde{C}_2)^{-1} (\tilde{C}_2^T A x - d_2 + \tilde{C}_2^T M z)). \tag{27}$$

Once again, $p_2 + 1$ simulations are required, because computing $\tilde{C}_2$ requires p_2 solves of A^{-T} and computing Δx requires 1 solve of A^{-1}.

4.1.4 Combined performance objective

As can be seen from the previous examples, efficiently computing Δx involves transforming $Ax \to z$, and then converting back, $A^{-1} z \to x$. It should be no surprise then, that we can also include all three performance objectives in a single $f(x)$.

This results in computing

$$\Delta x = -A^{-1} M (z - \tilde{C}_2 (\tilde{C}_2^T M \tilde{C}_2)^{-1} (\tilde{C}_2^T A x - d_2 + \tilde{C}_2^T M z)), \tag{28}$$

where

$$M = I - \tilde{C}_1 (\rho I + \tilde{C}_1^T \tilde{C}_1)^{-1} \tilde{C}_1^T \tag{29a}$$

$$\begin{aligned} z &= A^{-T} \nabla_x \mathcal{L} \\ &= \rho((I + \frac{1}{\rho} \tilde{C}_1 \tilde{C}_1^T) A x - b + \frac{1}{\rho}(y + \tilde{c} - \tilde{C}_1 d_1)) \end{aligned} \tag{29b}$$

$$\tilde{c} = A^{-1} c \tag{29c}$$

$$\tilde{C}_1 = A^{-1} C_1 \tag{29d}$$

$$\tilde{C}_2 = A^{-1} C_2. \tag{29e}$$

As expected, the first transformation costs $1 + p_1 + p_2$ solves of A^{-T}, and the second transformation costs 1 solve of A^{-1}.

4.1.5 Concave, low-rank quadratic $f(x)$

As a special case, we now consider the choice of $f(x) = -\frac{1}{2}\|C_1^T x\|^2$, which is a quadratic, but concave, performance objective. We consider the case where $f(x)$ is low-rank, that is, $C_1 \in \mathbf{R}^{n \times p_1}$ where $p_1 \ll n$.

To deal with the concavity of $f(x)$ we would like to use a truncated Newton method,

$$\hat{\mathcal{L}}(x, p, y) = \mathcal{L}(x, p, y) + \frac{\eta}{2}\|x - x_0\|^2 \tag{30}$$

or even

$$\Delta x = -(\nabla^2_{xx}\mathcal{L} + \eta I)^{-1}\nabla_x \mathcal{L}, \tag{31}$$

where the coefficient η is increased in order to ensure that the augmented Hessian is positive definite (so that $(\nabla_x \mathcal{L})^T \Delta x > 0$). However, this generally makes inverting the Hessian intractable.

A computationally efficient alternative would be to augment $\mathcal{L}(x, p, y)$ in the following way,

$$\hat{\mathcal{L}}(x, p, y) = \mathcal{L}(x, p, y) + \frac{\eta}{2}\|A(x - x_0)\|^2 \tag{32}$$

so that

$$(\nabla^2_{xx}\hat{\mathcal{L}})^{-1} = A^{-1}(\rho I + \eta I - \tilde{C}_1 \tilde{C}_1^T)^{-1} A^{-T} \tag{33}$$

where $\tilde{C}_1 = A^{-T} C_1$ and $\nabla^2_{xx}\mathcal{L} = \rho A^T A$.

For the case $p_1 = 1$, positive definiteness can be ensured by choosing $\eta = \|\tilde{C}_1\|^2$. Generally, for $p_1 > 1$, η can be set to the extremal eigenvalue of $\tilde{C}_1 \tilde{C}_1^T$ in order to ensure positive definiteness. Efficient computation of such an eigenvalue can be performed via a QR factorization of $\tilde{C}_1$.

The computational cost of using a truncated Newton method is that multiple steps are needed, and so Δx must be computed many times. Fortunately, $\tilde{C} = A^{-T} C$ needs only to be computed once, bringing the total number of solves to $n_{tn} + p_1$, where n_{tn} is the number of steps needed for the truncated Newton method.

4.1.6 Linearly constrained $f(x)$ with arbitrary phase

We now investigate another special case of $f(x)$, namely, when $f(x)$ implements the equality constraint $C_2^T x = e^{i\phi} d_2$, where $C_2 \in \mathbf{R}^{n \times p_2}$ and $p_2 \ll n$. This situation corresponds to the addition of an arbitrary phase $e^{i\phi}$ to a linear equality constraint.

We simply note that this case can be handled by computing

$$\Delta x = -A^{-1} M (z - \tilde{C}_2 (\tilde{C}_2^T M \tilde{C}_2)^{-1} (\tilde{C}_2^T A x - e^{i\phi} d_2 + \tilde{C}_2^T M z)). \tag{34}$$

for different choices of ϕ. The total computational cost is then simply $n_\phi + p_2$ solves, where n_ϕ is the number of values of ϕ which are tried.

4.2 Multi-mode ADMM

Since ADMM explicitly separates updating the field and structure variables, handling multiple fields is very naturally included in the formulation. Namely, multiple fields (x_i) are updated simultaneously, in parallel, and then care "connected" by updating a single p which takes all the x_i into account.

A Constructing the relevant matrices and vectors

B Solving the matrices

C Linear algebra definitions

C.1 Gradient and Hessian calculations

C.2 Matrix inversion lemma

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1} \tag{35}$$

C.3 Block elimination of a matrix

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{36}$$

$$A_{11}x_1 + A_{12}x_2 = b_1 \tag{37a}$$

$$x_1 = A_{11}^{-1}(b_1 - A_{12}x_2) \tag{37b}$$

$$A_{21}x_1 + A_{22}x_2 = b_2 \tag{38a}$$

$$A_{21}A_{11}^{-1}(b_1 - A_{12}x_2) + A_{22}x_2 = b_2 \tag{38b}$$

$$(A_{22} - A_{21}A_{11}^{-1}A_{12})x_2 = b_2 - A_{21}A_{11}^{-1}b_1 \tag{38c}$$

Typically, one computes $A_{11}^{-1}A_{12}$ and $A_{11}^{-1}b_1$, and then solves for the rest of the system.

References

[1] Boyd group ADMM paper