PDE-constrained optimization and the adjoint method[1]

Andrew M. Bradley July 7, 2024[2] (original November 16, 2010)

PDE-constrained optimization and the adjoint method for solving these and related problems appear in a wide range of application domains. Often the adjoint method is used in an application without explanation. The purpose of this tutorial is to explain the method in detail in a general setting that is kept as simple as possible.

We use the following notation: the total derivative (gradient) is denoted d_x (usually denoted $\mathrm{d}(\cdot)/\mathrm{d}x$ or ∇_x); the partial derivative, ∂_x (usually, $\partial(\cdot)/\partial_x$); the differential, d. We also use the notation f_x for both partial and total derivatives when we think the meaning is clear from context. Recall that a gradient is a row vector, and this convention induces sizing conventions for the other operators. We use only real numbers in this presentation.

1 The adjoint method

Let $x \in \mathbb{R}^{n_x}$ and $p \in \mathbb{R}^{n_p}$. Suppose we have the function $f(x,p) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}$ and the relationship $g(x,p) = 0$ for a function $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ whose partial derivative g_x is everywhere nonsingular. What is $\mathrm{d}_p f$?

1.1 Motivation

The equation $g(x,p) = 0$ is often solved by a complicated software program that implements what is sometimes called a *simulation* or the *forward problem*. Given values for the parameters p, the program computes the values x. For example, p could parameterize boundary and initial conditions and material properties for a discretized PDE, and x are the resulting field values. $f(x,p)$ is often a measure of merit: for example, fit of x to data at a set of locations, the smoothness of x or p, the degree to which p attains a particular objective. Minimizing f is sometimes called the *inverse problem*.

The gradient $\mathrm{d}_p f$ is useful in many contexts: for example, to solve the optimization problem $\min_p f$ or to assess the sensitivity of f to the elements of p.

One method to approximate $\mathrm{d}_p f$ is to compute n_p finite differences over the elements of p. Each finite difference computation requires solving $g(x,p) = 0$. For moderate to large n_p, this can be quite costly.

[1]This document is licensed under CC BY 4.0.

[2]Revisions: (2019) CC license, fix some typos. (2024) Reorganize Sec. 2; remove seismic tomography example.

In the program to solve $g(x,p) = 0$, it is likely that the Jacobian matrix $\partial_x g$ is calculated (see Sections 1.3 and 1.5 for further details). The *adjoint method* uses the transpose of this matrix, g_x^T, to compute the gradient $\mathrm{d}_p f$. The computational cost is usually no greater than solving $g(x,p) = 0$ once and sometimes even less costly.

1.2 Derivation

In this section, we consider the slightly simpler function $f(x)$; see below for the full case.

First,

$$\mathrm{d}_p f = \mathrm{d}_p f(x(p)) = \partial_x f \mathrm{d}_p x \; (= f_x x_p). \tag{1}$$

Second,

$$\begin{aligned} g(x,p) = 0 &\text{ everywhere implies} \\ \mathrm{d}_p g &= 0. \end{aligned}$$

(Note carefully that $\mathrm{d}_p g = 0$ everywhere only because $g = 0$ everywhere. It is certainly not the case that a function that happens to be 0 at a point also has a 0 gradient there.) Expanding the total derivative,

$$g_x x_p + g_p = 0.$$

As g_x is everywhere nonsingular, the final equality implies $x_p = -g_x^{-1} g_p$. Substituting this latter relationship into (1) yields

$$\mathrm{d}_p f = -f_x g_x^{-1} g_p.$$

The expression $-f_x g_x^{-1}$ is a row vector times an $n_x \times n_x$ matrix and may be understood in terms of linear algebra as the solution to the linear equation

$$g_x^T \lambda = -f_x^T, \tag{2}$$

where T is the matrix transpose. The matrix conjugate transpose (just the transpose when working with reals) is also called the matrix *adjoint*, and for this reason, the vector λ is called the vector of *adjoint variables* and the linear equation (2) is called the *adjoint equation*. In terms of λ, $\mathrm{d}_p f = \lambda^T g_p$.

A second derivation is useful. Define the *Lagrangian*

$$\mathcal{L}(x,p,\lambda) \equiv f(x) + \lambda^T g(x,p),$$

where in this context λ is the vector of *Lagrange multipliers*. As $g(x,p)$ is everywhere zero by construction, we may choose λ freely, $f(x) = \mathcal{L}(x,p,\lambda)$, and

$$\begin{aligned} \mathrm{d}_p f(x) = \mathrm{d}_p \mathcal{L} &= \partial_x f \mathrm{d}_p x + \mathrm{d}_p \lambda^T g + \lambda^T (\partial_x g \mathrm{d}_p x + \partial_p g) \\ &= f_x x_p + \lambda^T (g_x x_p + g_p) \quad \text{because } g = 0 \text{ everywhere} \\ &= (f_x + \lambda^T g_x) x_p + \lambda^T g_p. \end{aligned}$$

If we choose λ so that $g_x^T \lambda = -f_x^T$, then the first term is zero and we can avoid calculating x_p. This condition is the adjoint equation (2). What remains, as in the first derivation, is $\mathrm{d}_p f = \lambda^T g_p$.

1.3 The relationship between the constraint and adjoint equations

Suppose $g(x,p) = 0$ is the linear (in x) equation $A(p)x - b(p) = 0$. As $\partial_x g = A(p)$, the adjoint equation is $A(p)^T \lambda = -f_x^T$. The two equations differ in form only by the adjoint.

If $g(x,p) = 0$ is a nonlinear equation, then software that solves the system for x given a particular value for p quite likely solves, at least approximately, a sequence of linear equations of the form

$$\partial_x g(x,p) \Delta x = -g(x,p). \tag{3}$$

$\partial_x g = g_x$ is the Jacobian matrix for the function $g(x,p)$, and (3) is the linear system that gives the step to update x in Newton's method. The adjoint equation $g_x^T \lambda = -f_x^T$ solves a linear system that differs in form from (3) only by the adjoint operation.

1.4 f is a function of both x and p

Suppose our function is $f(x,p)$ and we still have $g(x,p) = 0$. How does this change the calculations? As

$$\mathrm{d}_p f = f_x x_p + f_p = \lambda^T g_p + f_p,$$

the calculation changes only by the term f_p, which usually is no harder to compute in terms of computational complexity than f_x.

f directly depends on p, for example, when the modeler wishes to weight or penalize certain parameters. For example, suppose f originally measures the misfit between simulated and measured data; then f depends directly only on x. But suppose the model parameters p vary over space and the modeler prefers smooth distributions of p. Then a term can be added to f that penalizes nonsmooth p values.

1.5 Partial derivatives

We have seen that $\partial_x g$ is the Jacobian matrix for the nonlinear function $g(x,p)$ for fixed p. To obtain the gradient $\mathrm{d}_p f$, $\partial_p g$ is also needed. This quantity generally is no harder to calculate than g_x. But it will almost certainly require writing additional code, as the original software to solve just $g(x,p) = 0$ does not require it.

2 PDE-constrained optimization problems

Partial differential equations are used to model physical processes. Optimization over a PDE arises in at least two broad contexts: determining parameters of a PDE-based model so that the field values match observations (an *inverse problem*); and design optimization: for example, of an airplane wing.

A common, straightforward, and very successful approach to solving PDE-constrained optimization problems is to solve the numerical optimization problem resulting from discretizing the PDE. Such problems take the form

$$\begin{aligned} &\underset{p}{\text{minimize}} \quad f(x,p) \\ &\text{subject to} \quad g(x,p) = 0. \end{aligned}$$

An alternative is to discretize the first-order optimality conditions corresponding to the original problem; this approach has been explored in various contexts for theoretical reasons but generally is much harder and is not as practically useful a method.

Two broad approaches solve the numerical optimization problem. The first approach is that of modern, cutting-edge optimization packages: converge to a feasible solution ($g(x,p) = 0$) only as f converges to a minimizer. The second approach is to require that x be feasible at every step in p ($g(x,p) = 0$).

The first approach is almost certainly the better approach for almost all problems. However, practical considerations turn out to make the second approach the better one in many applications. For example, a research effort may have produced a complicated program to solve $g(x,p) = 0$ (the PDE or *forward problem*), and one now wants to solve an optimization problem (*inverse problem*) using this existing code. Additionally, other properties of particularly time-dependent problems can make the first approach very difficult to implement.

In the second approach, the problem solver must evaluate $f(x,p)$, solve $g(x,p) = 0$, and provide the gradient $\mathrm{d}_p f$. Section 1 provides the necessary tools at a high level of generality to perform the final step. But at least one class of problems deserves some additional discussion.

2.1 Time-dependent problems

Time-dependent problems have special structure for two reasons. First, the matrices of partial derivatives have very strong block structure; we shall not discuss this low-level topic here. Second, and the subject of this section, time-dependent problems are often treated by *semi-discretization*: the spatial derivatives are made explicit in the various operators, but the time integration is treated as being continuous; this *method of lines* induces a system of ODE. The method-of-lines treatment has two implications. First, the adjoint equation for the problem is also an ODE induced by the method of lines, and the derivation of the adjoint equation must reflect that. Second, the forward and adjoint ODE can be solved by standard adaptive ODE integrators.

2.1.1 The adjoint method for the first-order problem

Consider the problem

$$\begin{aligned} \underset{p}{\text{minimize}} \quad & F(x,p), \quad \text{where} \quad F(x,p) \equiv \int_0^T f(x,p,t)\,\mathrm{d}t, \\ \text{subject to} \quad & h(x,\dot{x},p,t) = 0 \\ & g(x(0),p) = 0, \end{aligned} \tag{4}$$

where p is a vector of unknown parameters; x is a (possibly vector-valued) function of time; $h(x,\dot{x},p,t)=0$ is an ODE in implicit form; and $g(x(0),p)=0$ is the initial condition, which is a function of some of the unknown parameters. The ODE h may be the result of *semi-discretizing* a PDE, which means that the PDE has been discretized in space but not time. An ODE in explicit form appears as $\dot{x} = \bar{h}(x,p,t)$, and so the implicit form is $h(x,\dot{x},p,t) = \dot{x} - \bar{h}(x,p,t)$.

A gradient-based optimization algorithm requires the user to calculate the total derivative (gradient)

$$\mathrm{d}_p F(x,p) = \int_0^T [\partial_x f\ \mathrm{d}_p x + \partial_p f]\ \mathrm{d}t.$$

Calculating $\mathrm{d}_p x$ is difficult in most cases. As in Section 1, two common approaches simply do away with having to calculate it. One approach is to approximate the gradient $\mathrm{d}_p F(x,p)$ by finite differences over p. Generally, this requires integrating n_p additional ODE. The second method is to develop a second ODE, this one in the adjoint vector λ, that is instrumental in calculating the gradient. The benefit of the second approach is that the total work of computing F and its gradient is approximately equivalent to integrating only two ODE.

The first step is to introduce the Lagrangian corresponding to the optimization problem:

$$\mathcal{L} \equiv \int_0^T [f(x,p,t) + \lambda^T h(x,\dot{x},p,t)]\ \mathrm{d}t + \mu^T g(x(0),p).$$

The vector of Lagrangian multipliers λ is a function of time, and μ is another vector of multipliers that are associated with the initial conditions. Because the two constraints $h=0$ and $g=0$ are always satisfied by construction, we are free to set the values of λ and μ, and $\mathrm{d}_p\mathcal{L} = \mathrm{d}_p F$. Taking this total derivative,

$$\begin{aligned} \mathrm{d}_p\mathcal{L} = & \int_0^T [\partial_x f \mathrm{d}_p x + \partial_p f + \lambda^T(\partial_x h \mathrm{d}_p x + \partial_{\dot{x}} h \mathrm{d}_p \dot{x} + \partial_p h)]\ \mathrm{d}t \\ & + \mu^T(\partial_{x(0)} g\ \mathrm{d}_p x(0) + \partial_p g). \end{aligned} \tag{5}$$

The integrand contains terms in $\mathrm{d}_p x$ and $\mathrm{d}_p \dot{x}$. The next step is to integrate by parts to eliminate the second one:

$$\int_0^T \lambda^T \partial_{\dot{x}} h\ \mathrm{d}_p \dot{x}\ \mathrm{d}t = \lambda^T \partial_{\dot{x}} h\ \mathrm{d}_p x\big|_0^T - \int_0^T [\dot{\lambda}^T \partial_{\dot{x}} h + \lambda^T \mathrm{d}_t \partial_{\dot{x}} h]\ \mathrm{d}_p x\ \mathrm{d}t. \tag{6}$$

Substituting this result into (5) and collecting terms in $\mathrm{d}_p x$ and $\mathrm{d}_p x(0)$ yield

$$\begin{aligned} \mathrm{d}_p\mathcal{L} = & \int_0^T [(\partial_x f + \lambda^T(\partial_x h - \mathrm{d}_t \partial_{\dot{x}} h) - \dot{\lambda}^T \partial_{\dot{x}} h)\mathrm{d}_p x + f_p + \lambda^T \partial_p h]\ \mathrm{d}t \\ & + \lambda^T \partial_{\dot{x}} h\ \ \mathrm{d}_p x\big|_T + (-\lambda^T \partial_{\dot{x}} h + \mu^T g_{x(0)})\big|_0 \mathrm{d}_p x(0) + \mu^T g_p. \end{aligned}$$

As we have already discussed, $\mathrm{d}_p x(T)$ is difficult to calculate. Therefore, we set $\lambda(T) = 0$ so that the whole term is zero. Similarly, we set $\mu^T = \lambda^T \partial_{\dot{x}} h|_0 g_{x(0)}^{-1}$ to cancel the second-to-last term. Finally, we can avoid computing $\mathrm{d}_p x$ at all other times $t > 0$ by setting

$$\partial_x f + \lambda^T(\partial_x h - \mathrm{d}_t \partial_{\dot{x}} h) - \dot{\lambda}^T \partial_{\dot{x}} h = 0.$$

The algorithm for computing $\mathrm{d}_p F$ follows:

1. Integrate $h(x,\dot{x},p,t) = 0$ for x from $t=0$ to T with initial conditions $g(x(0),p)=0$.
2. Integrate $\partial_x f + \lambda^T(\partial_x h - \mathrm{d}_t \partial_{\dot{x}} h) - \dot{\lambda}^T \partial_{\dot{x}} h = 0$ for λ from $t=T$ to 0 with initial conditions $\lambda(T)=0$.
3. Set

$$\mathrm{d}_p F = \int_0^T [f_p + \lambda^T \partial_p h]\ \mathrm{d}t + \lambda^T \partial_{\dot{x}} h\big|_0 g_{x(0)}^{-1} g_p.$$

2.1.2 The relationship between the constraint and adjoint equations

Suppose $h(x, \dot{x}, p, t)$ is the first-order explicit linear ODE $h = \dot{x} - A(p)x - b(p)$. Then $h_x = -A(p)$ and $h_{\dot{x}} = I$, and so the adjoint equation is $f_x - \lambda^T A(p) - \dot{\lambda}^T = 0$. The adjoint equation is solved backward in time from T to 0. Let $\tau \equiv T - t$; hence $\mathrm{d}_t = -\mathrm{d}_\tau$. Denote the total derivative with respect to τ by a prime. Rearranging terms in the two equations,

$$\begin{aligned}\dot{x} &= A(p)x + b(p)\\ \dot{\lambda} &= A(p)^T\lambda - f_x^T.\end{aligned}$$

The equations differ in form only by an adjoint.

2.1.3 A simple closed-form example

As an example, let's calculate the gradient of

$$\begin{aligned}&\int_0^T x \ \mathrm{d}t\\ \text{subject to } \ &\dot{x} = bx\\ &x(0) - a = 0.\end{aligned}$$

Here, $p = [a\ b]^T$ and $g(x(0), p) = x(0) - a$. We follow each step:

1. Integrating the ODE yields $x(t) = ae^{bt}$.
2. $f(x, p, t) \equiv x$ and so $\partial_x f = 1$. Similarly, $h(x, \dot{x}, p, t) \equiv \dot{x} - bx$, and so $\partial_x h = -b$ and $\partial_{\dot{x}} h = 1$. Therefore, we must integrate

$$\begin{aligned}1 - b\lambda - \dot{\lambda} &= 0\\ \lambda(T) &= 0,\end{aligned}$$

which yields $\lambda(t) = b^{-1}(1 - e^{b(T-t)})$.

3. $\partial_p f = [0\ 0]$, $\partial_p h = [0\ \ -x]$, $g_{x(0)} = 1$, and $g_p = [-1\ 0]$. Therefore, the first component of the gradient is

$$\lambda^T \partial_{\dot{x}} h\big|_0 g_{x(0)}^{-1} g_p = \lambda(0) \cdot 1 \cdot 1^{-1} \cdot (-1) = b^{-1}(-1 + e^{bT});$$

and as $\partial_b g = 0$,

$$\int_0^T -\lambda x \ \mathrm{d}t = \int_0^T b^{-1}(e^{b(T-t)} - 1)ae^{bt} \ \mathrm{d}t = \frac{a}{b}Te^{bT} - \frac{a}{b^2}(e^{bT} - 1)$$

is the second component.

As a check, let us calculate the total derivative directly. The objective is

$$\int_0^T x \ \mathrm{d}t = \int_0^T ae^{bt} \ \mathrm{d}t = \frac{a}{b}(e^{bT} - 1).$$

Taking the derivative of this expression with respect to a and, separately, b yields the same results we obtained by the adjoint method.

2.1.4 The adjoint method for the second-order problem

The derivation in this section follows that in Section 2.1.1 for the first-order problem. For simplicity, we assume the ODE can be written in explicit form. The general problem is

$$\begin{aligned}&\underset{m}{\text{minimize}} \ \ F(s, m), \quad \text{where} \quad F(s, m) \equiv \int_0^T f(s, m, t) \ \mathrm{d}t,\\ &\text{subject to } \ \ddot{s} = \bar{h}(s, m, t)\\ &\qquad\qquad\ g(s(0), m) = 0\\ &\qquad\qquad\ k(\dot{s}(0), m) = 0.\end{aligned} \tag{7}$$

The corresponding Lagrangian is

$$\mathcal{L} \equiv \int_0^T [f(s, m, t) + \lambda^T(\ddot{s} - \bar{h}(s, m, t))] \ \mathrm{d}t + \mu^T g(s(0), m) + \eta^T k(\dot{s}(0), m), \tag{8}$$

which differs from the Lagrangian for the first-order problem by the term for the additional initial condition and the simplified form of h. The total derivative is

$$\begin{aligned}\mathrm{d}_m\mathcal{L} = &\int_0^T [\partial_s f \mathrm{d}_m s + \partial_m f + \lambda^T(\mathrm{d}_m\ddot{s} - \partial_s\bar{h}\mathrm{d}_m s - \partial_m\bar{h})] \ \mathrm{d}t\\ &+ \mu^T(\partial_{s(0)} g \ \mathrm{d}_m s(0) + \partial_m g) + \eta^T(\partial_{\dot{s}(0)} k \ \mathrm{d}_m\dot{s}(0) + \partial_m k).\end{aligned}$$

Integrating by parts twice,

$$\int_0^T \lambda^T \mathrm{d}_m\ddot{s} \ \mathrm{d}t = \lambda^T \mathrm{d}_m\dot{s}\Big|_0^T - \dot{\lambda}^T \mathrm{d}_m s\Big|_0^T + \int_0^T \ddot{\lambda}^T \mathrm{d}_m s \ \mathrm{d}t.$$

Substituting this and grouping terms,

$$\begin{aligned}\mathrm{d}_m\mathcal{L} = &\int_0^T [(f_s - \lambda^T\bar{h}_s + \ddot{\lambda}^T)\mathrm{d}_m s + f_m - \lambda^T\bar{h}_m]\mathrm{d}t && \Rightarrow f_s - \lambda^T\bar{h}_s + \ddot{\lambda}^T = 0\\ &+ (\mu^T g_{s(0)} + \dot{\lambda}^T)|_0 \ \mathrm{d}_m s(0) && \Rightarrow \mu^T = -\dot{\lambda}(0)^T g_{s(0)}^{-1}\\ &+ (\eta^T k_{\dot{s}(0)} - \lambda^T)|_0 \ \mathrm{d}_m\dot{s}(0) && \Rightarrow \eta^T = \lambda(0)^T k_{\dot{s}(0)}^{-1}\\ &+ \lambda^T \mathrm{d}_m\dot{s}|_T && \Rightarrow \lambda(T) = 0\\ &- \dot{\lambda}^T \mathrm{d}_m s|_T && \Rightarrow \dot{\lambda}(T) = 0\\ &+ \mu^T g_m + \eta^T k_m.\end{aligned}$$

We have indicated the suitable multiplier values to the right of each term. Putting everything together, the adjoint equation is

$$\begin{aligned} \ddot{\lambda} &= \bar{h}_s^T \lambda - f_s^T \\ \lambda(T) &= \dot{\lambda}(T) = 0 \end{aligned}$$

and the total derivative of F is

$$\mathrm{d}_m F = \mathrm{d}_m \mathcal{L} = \int_0^T [f_m - \lambda^T \bar{h}_m] \mathrm{d}t - \dot{\lambda}(0)^T g_{s(0)}^{-1} g_m + \lambda(0)^T k_{\dot{s}(0)}^{-1} k_m.$$

2.1.5 The continuous, rather than discretized, problem

For simplicity, suppose $\bar{h}(s, m, t) = A(m)s + b(m)$. So far we have viewed $A(m)$ as being a matrix that results from discretizing a PDE. This is often the best way to view the adjoint problem in practice: the gradient of interest is that of the problem on the computer, which in general must be a discretized representation of the original continuous problem. However, it is still helpful to see how the adjoint method is applied to the fully continuous problem.

We shall continue to use the notation $A(m)$ to denote the spatial operator, but now we view it as something like $A(x; m) = \alpha(x; m)\nabla \cdot (\beta(x; m)\nabla)$, where α and β are functions of space x and parameterized by the model parameters m, and ∇ is the gradient operator.

The key difference between the discretized and continuous problems is the inner product between the Lagrange multiplier λ and the fields. In the discretized problem, we write $\lambda^T A(m)s$; in the continuous problem, we write $\int_\Omega \lambda(x)A(x; m)s(x)\ \mathrm{d}x$, where Ω is the domain over which the fields are defined. Here we assume $s(x)$ is a scalar field for clarity. Then the Lagrangian like (8) is

$$\begin{aligned} \mathcal{L} \equiv \int_0^T &\left[f(s, m, t) + \int_\Omega \lambda(\ddot{s} - \bar{h}(s, m, t))\ \mathrm{d}x \right]\ \mathrm{d}t\ + \\ &\int_\Omega [\mu g(s(0), m) + \eta k(\dot{s}(0), m)]\ \mathrm{d}x, \end{aligned}$$

In general, the derivations we have seen so far can be carried out with spatial integrals replacing the discrete inner products.

References

Y. Cao, S. Li, L. Petzold, R. Serban, "Adjoint sensitivity analysis for differential-algebraic equations: The adjoint DAE system and its numerical solution", *SIAM J. Sci. Comput.* (2003) 23(3), 1076–1089.