The Adjoint Problem and Its Applications

Mech 511

Version Date: March 9, 2020

The adjoint method uses an auxiliary PDE to determine the sensitivity of an output quantity to errors in the solution of the original PDE or to changes in design variables. This is a very powerful tool, used routinely for design, and increasingly for error estimation and adaptation.

1 The Continuous Adjoint Problem

Suppose we have some linear PDE, written as

$$Lu = f$$

where L is a linear partial differential operator, u is the solution of the PDE, and f is some source term.[1] We solve this on a domain Ω, subject to linear boundary conditions

$$Bu = e$$

on the boundary $\partial\Omega$, where B is some linear boundary differential operator, and the boundary conditions are inhomogeneous if $e \neq 0$. Also, note that for systems of equations, B may have a different dimension (different number of boundary conditions) on different parts of the boundary. We'll refer to this as the *primal problem.*

In addition, suppose that we have some *functional*[2] that we compute from the solution. The general form of this functional (in 2D) is

$$\begin{aligned} J &= \iint_\Omega u\, g\, dA + \oint_{\partial\Omega} h\, Cu\, ds \\ &= (g, u)_\Omega + (h, Cu)_{\partial\Omega} \end{aligned} \tag{1}$$

Here, J is our output functional. Cu is an algebraic and/or differential operator defined on the boundary; for instance, C could evaluate the pressure in a flow, or the shear stress, or the heat flux. g and h are weight functions; either of these may be zero over all or part of

[1] This description follows closely the work of Pierce and Giles [8]. This paper is one of several co-written by Michael Giles that lay out all of the theory for these methods in detail [1, 2, 5, 6, 3, 4, 7]. Unfortunately, none of them taken individually is complete.

[2] A functional is a value that depends on some function.

the domain (the boundary of the domain, for h). The second line defines compact notation for an integral inner product $(\cdot,\cdot)_\Omega$ over the domain and another $(\cdot,\cdot)_{\partial\Omega}$ over the boundary.

The corresponding adjoint problem on this same domain is

$$L^*v = g$$

subject to the boundary condition

$$B^*v = h$$

We define these operators, as well as C^* by the adjoint identity

$$(v, Lu)_\Omega + (C^*v, Bu)_{\partial\Omega} = (L^*v, u)_\Omega + (B^*v, Cu)_{\partial\Omega}$$

The right side of this identity is the primal output functional, and the left side is the dual output functional. The adjoint operators L^*, B^*, and C^* are defined from primal operators by integration by parts. The adjoint problem is also referred to as the *dual problem.*

Using the adjoint identity and the definition of the primal problem, we can re-write the output functional J as

$$\begin{aligned} J &= (g, u)_\Omega + (h, Cu)_{\partial\Omega} \\ &= (v, f)_\Omega + (C^*v, e)_{\partial\Omega} \end{aligned} \tag{2}$$

Eq. 2 says that there are two exactly equivalent ways to calculate the functional, which should give identical answers. Discrete formulations for which this is exactly true are called *adjoint consistent.* Formulations for which this is true only in the limit as you refine the mesh are referred to as *asymptotically adjoint consistent.* Formulations for which Eq. 2 does not hold are called wrong.

1.1 An Example

Suppose that our primal problem is a Poisson problem on $(x, y) \in [0, 1] \times [0, 1]$:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{3\pi^2}{4} \frac{\cos\left(\frac{\pi x}{2}\right) \sin h\left(\pi y\right)}{\sinh \pi}$$

with boundary conditions:

$$\begin{aligned} u &= 0 \qquad \text{on } x = 1 \text{ and } y = 0 \\ u &= \cos\frac{\pi x}{2} \qquad \text{on } y = 1 \\ \frac{\partial u}{\partial x} &= 0 \qquad \text{on } x = 0 \end{aligned}$$

The exact solution of this PDE is

$$u = \frac{\cos\left(\frac{\pi x}{2}\right) \sinh\left(\pi y\right)}{\sinh \pi}$$

Define an output functional J as above, with $g = \sin(\pi x)\sin(\pi y)$. On the bottom boundary ($y = 0$), $Cu = \frac{\partial u}{\partial n} = -\frac{\partial u}{\partial y}$ and $h = \sin(\pi x)$; on the left boundary, $Cu = u$ and $h = \frac{\sin \pi y}{2\pi} - \frac{\pi \sinh(\pi(1-y))}{\sinh \pi}$; on the other two boundaries, $h = 0$. So our functional can be written as:

$$\begin{aligned}
J =& (u, g)_\Omega + (Cu, h)_{\partial\Omega} \\
=& \int_0^1 \int_0^1 \frac{\cos\left(\frac{\pi x}{2}\right)\sinh(\pi y)}{\sinh \pi} (\sin(\pi x)\sin(\pi y))\, dx\, dx \\
&+ \int_0^1 -\frac{\cos\left(\frac{\pi x}{2}\right)\pi}{\sinh \pi} \sin \pi x\, dx + \int_1^0 uh\, dy \\
=& \frac{2}{3\pi^2} - \frac{4}{3\sinh \pi} + \left\{ -\frac{1}{2\sinh \pi} - \frac{1}{4\pi^2} + \frac{1}{2}\frac{\pi \cosh \pi}{\sinh^2 \pi} \right\} \\
=& \frac{5}{12\pi^2} - \frac{11}{6\sinh \pi} + \frac{1}{2}\frac{\pi \cosh \pi}{\sinh^2 \pi}
\end{aligned}$$

Now we'll derive the adjoint problem, and show that the output functionals are identical. Before starting in on the adjoint identity, we need to formally write out definitions for Bu, e, Cu, and h

$$Bu = \begin{cases} u & x = 1, y = 0, y = 1 \\ \frac{\partial u}{\partial x} & x = 0 \end{cases}$$

$$e = \begin{cases} 0 & x = 1, y = 0 \\ \cos \frac{\pi x}{2} & y = 1 \\ 0 & x = 0 \end{cases}$$

$$Cu = \begin{cases} \frac{\partial u}{\partial y} & y = 0 \\ u & x = 0, x = 1, y = 1 \end{cases}$$

$$h = \begin{cases} \sin \pi x & y = 0 \\ -\frac{\sin \pi y}{2} + \frac{\pi \sinh(\pi(y-1))}{\sinh \pi} & x = 0 \\ 0 & x = 1, y = 1 \end{cases}$$

Now, repeating the adjoint identity:

$$(v, Lu)_\Omega + (C^* v, Bu)_{\partial\Omega} = (L^* v, u)_\Omega + (B^* v, Cu)_{\partial\Omega}$$

Focusing first on the area integral term, we can write, using integration by parts,

$$\begin{aligned}
\iint v\,(\nabla \cdot \nabla u)\, dx\, dy &= -\iint \nabla v \cdot \nabla u\, dx\, dy + \oint v \frac{\partial u}{\partial n}\, ds \\
&= \iint (\nabla \cdot \nabla v)\, u\, dx\, dy - \oint u \frac{\partial v}{\partial n}\, ds + \oint v \frac{\partial u}{\partial n}\, ds \qquad (3)
\end{aligned}$$

The term on the LHS is of course $(v, Lu)_\Omega$, and the first term on the RHS is $(L^* v, u)_\Omega$. The other two terms will help us define C^* and B^*.

Let's divide the domain boundary $\partial\Omega$ into a part $\partial\Omega_D$ where the primal problem has a Dirichlet boundary condition, and a part $\partial\Omega_N$ where the primal problem has a Neumann

boundary condition. Then we can split the boundary terms in Eq. 3 into Dirichlet and Neumann pieces:

$$\left(v, \nabla^2 u\right)_\Omega + \left(u, \frac{\partial v}{\partial n}\right)_{\partial\Omega_D} + \left(\frac{\partial u}{\partial n}, -v\right)_{\partial\Omega_N} = \left(u, \nabla^2 v\right)_\Omega + \left(v, \frac{\partial u}{\partial n}\right)_{\partial\Omega_D} + \left(u, -\frac{\partial v}{\partial n}\right)_{\partial\Omega_N}$$

Comparing this to the adjoint identity, and using the definitions of B and C, we get that

$$\begin{aligned} B^* v &= v \qquad C^* v = \frac{\partial v}{\partial n} \qquad on\, \partial\Omega_D \\ B^* v &= -\frac{\partial v}{\partial n} \qquad C^* v = -v \qquad on\, \partial\Omega_N \end{aligned}$$

In the end, the dual problem can be stated as:

$$\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} = \sin(\pi x)\sin(\pi y)$$

subject to the boundary conditions

$$\begin{aligned} \frac{\partial v}{\partial x} = -\frac{\partial v}{\partial n} = -h = -\frac{\sin \pi y}{2\pi} + \frac{\pi \sinh(\pi(1-y))}{\sinh \pi} \qquad x = 0 \\ v = \sin \pi x \qquad y = 0 \\ v = 0 \qquad x = 1, y = 1 \end{aligned}$$

and the dual functional is given by

$$J = (v, f)_\Omega + (C^* v, e)_{\partial\Omega}$$

The exact solution for the dual problem can be found by superposition: one term that satisfies $\nabla^2 v = \sin \pi x \sin \pi y$ with homogeneous Dirichlet boundary conditions, and a second term that satisfies $\nabla^2 v = 0$ with homogeneous Dirichlet boundary conditions on three sides and $v = \sin \pi x$ on $y = 0$. In this case, $h(x = 0, y)$ was set to match the actual gradient of v there; since the corresponding primal boundary integral term is zero anyway, this has no effect on the primal problem. In the end,

$$\begin{aligned} v = &-\frac{\sin(\pi x)\sin(\pi y)}{2\pi^2} \\ &+ \frac{\sin(\pi x)\sinh(\pi(1-y))}{\sinh \pi} \end{aligned}$$

and when we evaluate the functional, we get:

$$\begin{aligned} J &= (v, f)_\Omega + (C^* v, e)_{\partial\Omega} \\ &= \int_0^1 \int_0^1 v \frac{4}{3\pi^2} \frac{\cos\left(\frac{\pi x}{2}\right) \sin h(\pi y)}{\sinh \pi}\, dx\, dy + \int_0^1 \frac{\partial v}{\partial y} \cos \frac{\pi x}{2}\, dx \\ &= \left\{ \frac{1}{2} \frac{\pi \cosh \pi}{\sinh^2 \pi} - \frac{1}{2 \sinh \pi} - \frac{1}{4\pi^2} \right\} + \left\{ -\frac{4}{3 \sinh \pi} + \frac{2}{3\pi^2} \right\} \\ &= \frac{5}{12\pi^2} - \frac{11}{6 \sinh \pi} + \frac{1}{2} \frac{\pi \cosh \pi}{\sinh^2 \pi} \end{aligned}$$

In the end, the two problems are equivalent, in that we can evaluate the functional to get exactly the same results either way. So far, this is interesting, but not terribly helpful: the primal and dual problems are equally difficult to solve, so the second formulation hasn't done us any good yet.

2 The Discrete Adjoint Problem

There's a second way to come up with this sensitivity estimate: the discrete adjoint formulation. In this case, suppose we have a discretized PDE, written as

$$R_h\left(U_h, \mathbf{x}\right) = 0$$

where U_h is the discrete solution, R_h is the residual, and $\mathbf{x}$ is some collection of problem parameters, including things like geometry and flow conditions. We also have a scalar output quantity $J_h\left(U_h, \mathbf{x}\right)$. The sensitivity of this output to changes in the parameters is

$$\frac{dJ_h}{d\mathbf{x}} = \frac{\partial J_h}{\partial U_h}\frac{\partial U_h}{\partial \mathbf{x}} + \frac{\partial J_h}{\partial \mathbf{x}} \tag{4}$$

The two partial derivatives of J_h on the RHS are typically easy to calculate: after all, we have a way to calculate J_h, and we can differentiate that. The hard term is $\frac{\partial U_h}{\partial \mathbf{x}}$. The key here is that we need to enforce that the residual is still zero; in other words, we're interested in the change in the output functional with a change in parameters *at the converged solution.* The zero residual condition can be written as

$$\begin{aligned}
\frac{dR_h}{d\mathbf{x}} &= \frac{\partial R_h}{\partial U_h}\frac{\partial U_h}{\partial \mathbf{x}} + \frac{\partial R_h}{\partial \mathbf{x}} = 0 \\
\frac{\partial U_h}{\partial \mathbf{x}} &= -\left(\frac{\partial R_h}{\partial U_h}\right)^{-1}\frac{\partial R_h}{\partial \mathbf{x}}
\end{aligned}$$

The dependence of R_h on the parameters $\mathbf{x}$ is generally reasonably straightforward to compute, at least by finite difference; this does *not* require a flow solve every time we compute a new value of R_h, so it's cheap(ish). And of course $\frac{\partial R_h}{\partial U_h}$ is the global Jacobian matrix that we're already familiar with. Substituting this back into Eq. 4, we get:

$$\frac{dJ_h}{d\mathbf{x}} = -\frac{\partial J_h}{\partial U_h}\left(\frac{\partial R_h}{\partial U_h}\right)^{-1}\frac{\partial R_h}{\partial \mathbf{x}} + \frac{\partial J_h}{\partial \mathbf{x}} \tag{5}$$

This result is computable, but there's a problem: for every different parameter x_i, we have to solve a linear system with our global Jacobian as the matrix. This is okay if you have one parameter, but suppose you have, say, 200 shape parameters in an optimization problem? For N solution variables (components of U_h) and m parameters (components of $\mathbf{x}$), the shapes of these matrices are:

$$\underbrace{\frac{dJ_h}{d\mathbf{x}}}_{1\times m} = -\underbrace{\frac{\partial J_h}{\partial U_h}}_{1\times N}\underbrace{\left(\frac{\partial R_h}{\partial U_h}\right)^{-1}}_{N\times N}\underbrace{\frac{\partial R_h}{\partial \mathbf{x}}}_{N\times m} + \underbrace{\frac{\partial J_h}{\partial \mathbf{x}}}_{1\times m}$$

To get around this, we take the transpose of that Eq. 5, to get:

$$\underbrace{\frac{dJ_h}{d\mathbf{x}}^T}_{m\times 1} = -\underbrace{\frac{\partial R_h}{\partial \mathbf{x}}^T}_{m\times N}\underbrace{\left(\frac{\partial R_h}{\partial U_h}\right)^{-T}}_{N\times N}\underbrace{\frac{\partial J_h}{\partial U_h}^T}_{N\times 1} + \underbrace{\frac{\partial J_h}{\partial \mathbf{x}}^T}_{m\times 1} \tag{6}$$

Now we only have to solve a single linear system

$$\left(\frac{\partial R_h}{\partial U_h}\right)^T \psi = -\frac{\partial J_h}{\partial U_h}^T$$

for ψ, which is the discrete adjoint solution. Then we can get the sensitivity of the output functional with respect to all of the parameters from

$$\underbrace{\frac{dJ_h}{d\mathbf{x}}^T}_{m\times 1} = \underbrace{\frac{\partial R_h}{\partial \mathbf{x}}^T}_{m\times N}\underbrace{\psi}_{N\times 1} + \underbrace{\frac{\partial J_h}{\partial \mathbf{x}}^T}_{m\times 1} \tag{7}$$

When we have only one (or a few) objective functions, and a lot of design variables, this is a big advantage.

For flow solvers that explicitly calculate the global Jacobian matrix, the discrete adjoint problem is really easy to solve: it's just a single linear system solution at the end of the process. For flow solvers with an explicit global Jacobian matrix, solving the discrete adjoint problem requires, essentially, writing code that differentiates each step in the solution process, and applying those in reverse order (remember, there's a transpose in there...). This requires more coding but a lot less memory, and so many people are willing to pay that price.

2.1 Comparison to the Continuous Adjoint Problem

The discrete and continuous adjoint solutions both tell us the same thing: the sensitivity of a functional J to changes in the problem and/or to changes in how accurately we solve the problem. With the continuous adjoint, we start with a continuous PDE for the adjoint problem, discretize that, and solve it. With the discrete adjoint, we start with the discretized primal problem, differentiate that, and solve a resulting linear system. If things go as planned, the discrete and continuous formulations give the same solution with mesh refinement.

Either form of the adjoint can be used for all applications of adjoints; the choice between the two is philosophical, to some extent. The rest of the choice is driven by practical matters: implementation, boundary conditions for the continuous adjoint, etc.

3 Output Error Estimation

Unfortunately, we typically don't know the exact solution to our PDE, so we can't evaluate either Eq. 1 or Eq. 2 directly. Instead, we have to compute an estimate based on our numerical solution.

Suppose that we have some continuous approximation u_h (respectively, v_h) to our discrete solution to the primal (resp., dual) PDE on a mesh with characteristic size h. It is essential in

what follows that u_h (resp., v_h) be smooth enough that $f_h \equiv Lu_h$ (resp., $g_h \equiv L^* v_h$) is square integrable; yes, those are the continuous partial differential operators, not discrete operators. This specifically disallows Dirac δ-functions in f_h and g_h, for instance. In turn, this means that for unstructured meshes, u_h and v_h cannot be the result of ordinary reconstruction; the discontinuities in the reconstructed solution at control volume boundaries violate this condition.

Note that $f_h \neq f$, because $u_h \neq u$. The residual error $f_h - f$ is a computable measure of how closely u_h matches u, in the sense of satisfying the original PDE.

Let us now consider the primal functional, taking into account for the moment only the interior term. We will make repeated use of linearity of the inner product, the definitions of the primal and dual PDEs, and the duality condition.

$$\begin{aligned}
(g,u)_\Omega &= (g,u_h)_\Omega - (g,u_h-u)_\Omega \\
&= (g,u_h)_\Omega - (g_h,u_h-u)_\Omega + (g_h-g,u_h-u)_\Omega \\
&= (g,u_h)_\Omega - (L^* v_h,u_h-u)_\Omega + (g_h-g,u_h-u)_\Omega \\
&= (g,u_h)_\Omega - (v_h,L(u_h-u))_\Omega + (g_h-g,u_h-u)_\Omega \\
&= (g,u_h)_\Omega - (v_h,f_h-f)_\Omega + (g_h-g,u_h-u)_\Omega
\end{aligned} \tag{8}$$

The first term on the right is computable output functional. The second term is a computable estimate of the error in the functional. For structured meshes, $f_h - f$ (the truncation error) is of the same order as $u_h - u$ (the discretization error), and so this correction is higher-order accurate. The last term is the error that remains after correction. Because it involves $u_h - u$, which we don't know and can't easily estimate, this term is not considered possible to compute accurately enough for a second correction.

What does this mean practically?

Suppose we are using a structured mesh, and we have second-order accurate solutions to both the primal and dual PDEs. We can accurately estimate f_h by applying a fourth-order operator to our second-order solution. After we compute and apply the correction, we will have a fourth-order accurate corrected functional.

For finite-volume schemes on unstructured meshes, things are harder, because it's exceedingly difficult to satisfy the smoothness condition. It's possible, with some effort [9], to get an improvement in the order of accuracy, but not always the full order you expected. Finite element methods are more successful in calculating these corrections easily.

3.1 Correction for Functionals with Boundary Integrals

For functionals with boundary integrals, we can apply the same approach, expanding the functional in the terms we would naturally compute from our solution; computable correction terms; and remaining error terms.

$$\begin{aligned}
(g,u)_\Omega + (B^* v, Cu)_{\partial\Omega} = {} & (g,u_h)_\Omega + (h,Cu_h)_{\partial\Omega} \\
& + (v_h, f-f_h)_\Omega + (C^* v_h, Bu - Bu_h) \\
& + (g-g_h, u-u_h)_\Omega + (C^*(v-v_h), Bu - Bu_h)
\end{aligned} \tag{9}$$

The first line of the RHS is the original discrete functional, the second line is the computable correction term, and the third line is the remaining error. Note that $Bu - Bu_h$ is a measure of

how well the continuous projection u_h of our discrete solution matches the analytic boundary condition.

3.2 Using the Adjoint for Adaptation

The Junior CFD Ninja looks at Eq. 9 and says, "Hey, I've got this error term [the second line of the RHS] that I can compute. I'm going to use that to drive adaptation, and make that error term small!" And this works. In fact, this has been shown through numerical experiments to essentially always give the fastest rate of convergence to the correct value of the computed functional [the first line of the RHS] in terms of degrees of freedom and/or work done.

However, the Senior CFD Ninja looks at what the Junior Ninja did and says, "Ah, but grasshopper, you *know* how to compute that term. Use it to correct the functional. Adapt on an estimate of the remaining error." The catch here is that you need solution error estimates for at least one of the primal and dual problems. We've already seen that the corrected functional converges more quickly than the uncorrected functional. If adaptation is aimed at reducing the remaining error by increasing resolution where that term is high, then you would expect extremely fast convergence of the adaptive process, which is what you actually get. More on this when we talk about adaptation.

References

[1] Michael Giles and Niles Pierce. Adjoint equations in CFD — Duality, boundary conditions and solution behaviour. In *Proceedings of the Thirteenth AIAA Computational Fluid Dynamics Conference*, 1997. AIAA-97-1850.

[2] Michael Giles and Niles Pierce. Improved lift and drag estimates using adjoint Euler equations. In *Proceedings of the Fourteenth AIAA Computational Fluid Dynamics Conference*, 1999.

[3] Michael Giles and Niles Pierce. Adjoint error correction for integral outputs. In Timothy J. Barth and Herman Deconinck, editors, *Error Estimation and Adaptive Discretization Methods in Computational Fluid Dynamics*, volume 25 of *Lecture Notes in Computational Science and Engineering*, pages 47–95. Springer-Verlag, Berlin, 2003.

[4] Michael Giles, Niles Pierce, and Endre Suli. Progress in adjoint error correction for integral functionals. *Computing and Visualization in Science*, 6(2–3):113–121, 2004.

[5] Michael Giles and Niles A. Pierce. An introduction to the adjoint approach to design. *Flow, Turbulence, and Combustion*, 65(3–4):393–415, 2000.

[6] Michael Giles and Endre Suli. Adjoint methods for pdes: *a posteriori* error analysis and postprocessing by duality. *Acta Numerica*, 11:145–236, 2002.

[7] Niles Pierce and Michael Giles. Adjoint recovery of superconvergent functionals from PDE approximations. *SIAM Review*, 42(2):247–264, June 2000.

[8] Niles Pierce and Michael Giles. Adjoint and defect error bounding and correction for functional estimates. *Journal of Computational Physics*, 200(2):769–794, 2004.

[9] Mahkame Sharbatdar and Carl F Ollivier-Gooch. Adjoint-based functional correction for unstructured mesh finite volume methods. *Journal of Scientific Computing*, 2017.