The Adjoint Approach to Design,
Data Assimilation and Error Analysis

Mike Giles
Oxford University Computing Laboratory
ICFD Workshop on Adjoint Methods
July 21st, 1998

Introduction to adjoint analysis

Overview

What I’ll cover:
- the linear algebra viewpoint
- the p.d.e. connection

Key point: in all cases we’re interested in
one or more functionals
- objective function and constraint
  functions in design optimisation
- mismatch with experimental data
  in data assimilation
- error in key functionals in error analysis

Introduction to adjoint analysis

Linear Theory

Want to evaluate $g^T u$ given that $Au = f$.

The dual form is to evaluate $v^T f$ where $A^T v = g$.

The equivalence comes from $v^T f = v^T Au = (A^T v)^T u = g^T u$,
or, alternatively, $g^T u = g^T (A^{-1} f) = (g^T A^{-1}) f = v^T f$.

Introduction to adjoint analysis

Linear Theory

Suppose we want the objective function for $p$ different $f$’s, and $m$ different $g$’s.

Choice:
- either do $p$ different primal calculations
- or do $m$ different dual calculations

Adjoint approach is much cheaper when $m \ll p$. 

Introduction to adjoint analysis
### Linear Theory

What do adjoint variables mean?

**Answer 1:** They give you the influence of an arbitrary source term on the functional of interest.

\[ Au = f \]

source term \( \rightarrow \) functional \( v^T f \)

### Nonlinear design / data assimilation

For both, problem is to minimise \( J(U) \) subject to \( N(U, \alpha) = 0 \).

For aerodynamic design, may have

- \( \alpha \) – geometric design variables
- \( J(U) \) – drag
- \( N(U) \) – discrete flow equations

For data assimilation, may have

- \( \alpha \) – perturbed initial conditions
- \( J(U) \) – mismatch between model and experimental data
- \( N(U) \) – discrete modelling equations

### Linear Theory

**Answer 2:** They are the functional value corresponding to Green’s functions.

Consider

\[ f_i = (\ldots, 0, \frac{1}{\bar{u}^i}, 0, \ldots)^T. \]

Then corresponding solution \( u_i \) is the discrete equivalent of a Green’s function and

\[ v^T f = v_i = g^T u_i. \]

### Nonlinear design / data assimilation

Minimise \( J(U) \), subject to \( N(U, \alpha) = 0 \).

For single \( \alpha \), can linearise about a base solution \( U_0 \) to get:

\[ \frac{dJ}{d\alpha} = g^T u, \quad Au = f \]

where

\[ u \equiv \frac{dU}{d\alpha}, \quad g^T = \frac{\partial J}{\partial U}, \quad A = \frac{\partial N}{\partial U}, \quad f = -\frac{\partial N}{\partial \alpha}. \]

For multiple \( \alpha \) each has different \( f \), but same \( g \).
Nonlinear design / data assimilation

Two drawbacks:

1) to add a 'hard' constraint $J_2(U) = 0$, we need

$$\frac{dJ_2}{d\alpha} = y_2^T u$$

which requires a second adjoint calc.

Additional 'hard' constraints require even more adjoint calculations.

Alternative is to use 'soft' constraints via penalties in objective function.

Nonlinear design / data assimilation

2) If the objective function is of a least-squares type,

$$J(U) = \frac{1}{2} \sum_n (p_n(U) - P_n)^2,$$

then

$$\frac{dJ}{d\alpha_i} = \sum_n \frac{\partial p}{\partial U} \frac{dU}{d\alpha_i} (p_n(U) - P_n),$$

and so

$$\frac{d^2 J}{d\alpha_i d\alpha_j} \approx \sum_n \left( \frac{\partial p}{\partial U} \frac{dU}{d\alpha_i} \right) \left( \frac{\partial p}{\partial U} \frac{dU}{d\alpha_j} \right).$$

Introduction to adjoint analysis

Thus, the direct linear perturbation approach gives the approximate Hessian matrix, leading to very rapid convergence for the optimisation iteration.

By contrast, the adjoint approach provides no information on the Hessian, so the best optimisation methods take more steps to converge.

Introduction to adjoint analysis

Back to the original linear problem, evaluate $g^T u$ subject to

$$Au = f,$$

and the dual problem to evaluate $v^T f$ subject to

$$A^T v = g.$$

Now suppose, we have approximate solutions $\bar{u}, \bar{v}.$
**Linear error analysis**

Then, we have
\[
g^T u = g^T \tilde{u} + g^T (u - \tilde{u}) \\
= g^T \tilde{u} + v^T A (u - \tilde{u}) \\
= g^T \tilde{u} + v^T A (u - \tilde{u}) + (v - \tilde{v})^T A (u - \tilde{u}) \\
= g^T \tilde{u} + v^T (f - A \tilde{u}) + (v - \tilde{v})^T A (u - \tilde{u})
\]

No obvious benefits in linear algebra (\(?\)), but generalisation to p.d.e.’s is useful in grid adaptation (to reduce computable error) and error correction (through evaluating error).

**The PDE connection**

Suppose one wants to know \((g, u)\) given that \(u\) satisfies the p.d.e.
\[
Lu = f,
\]
plus homogeneous b.c.’s.

The adjoint formulation is \((v, f)\) where
\[
L^* v = g,
\]
plus homogeneous adjoint b.c.’s.

The equivalence comes from
\[
(v, f) = (v, Lu) \overset{\text{def}}{=} (L^* v, u) = (g, u).
\]

**Example**

\[
Lu = \frac{du}{dx} - \epsilon \frac{d^2 u}{dx^2}, \quad u(0) = u(1) = 0.
\]

\[
(v, Lu) = \int_0^1 v \left( \frac{du}{dx} - \epsilon \frac{d^2 u}{dx^2} \right) dx
\]
\[
= \int_0^1 u \left( -\frac{dv}{dx} - \epsilon \frac{d^2 v}{dx^2} \right) dx
\]
\[
+ \left[ vu - \epsilon \frac{du}{dx} + \epsilon u \frac{dv}{dx} \right]_0^1
\]
\[
= \int_0^1 u \left( -\frac{dv}{dx} - \epsilon \frac{d^2 v}{dx^2} \right) dx + \left[ -\epsilon \frac{du}{dx} \right]_0^1.
\]
Example

Thus, to satisfy the adjoint identity, we need

$$L^* v = \frac{dv}{dx} - \varepsilon \frac{d^2 v}{dx^2},$$

and the adjoint b.c.'s must be

$$v(0) = v(1) = 0.$$
### One last issue

When approximating p.d.e.’s there are two options in adjoint analysis.
- Fully discrete approach: discretise original p.d.e., linearise discrete equations, and then use the transpose for the adjoint.
- ‘Continuous’ approach: linearise original p.d.e., construct adjoint p.d.e. and associated b.c.’s, and then discretise.

Not yet clear which is best overall.

### Fully discrete approach

Advantages:
- in design/data assimilation applications, get exact gradient of discretised objective function
- creation of adjoint program is a straightforward process, in principle
- transposed matrix has same eigenvalues as original linearised matrix, so standard iteration method is guaranteed to converge

Disadvantages:
- programming can be tedious (but one could use automatic differentiation software?)
- may have to store some linearisation matrices, leading to large memory requirements

### Continuous approach

Advantages:
- role of adjoint b.c.’s is clearer
- adjoint program is perhaps simpler

Disadvantages:
- computed gradient will be slightly inconsistent with discrete objective function, so optimisation will not converge fully

Still very much an open issue as to which approach is better; right now final choice seems to come down to personal preference!