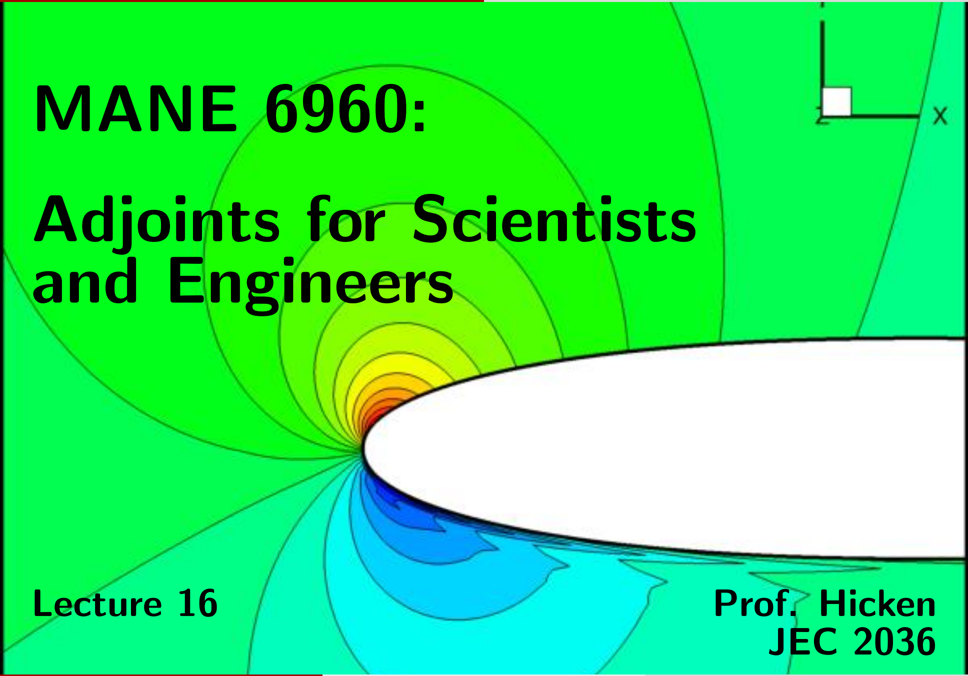


MANE 6960:

Adjoints for Scientists and Engineers

Lecture 16

Prof. Hicken
JEC 2036

Overview

This lecture covers a few subjects related to sensitivity analysis that do not fit neatly into any other lectures. Specifically, we will discuss

- how to compute accurate sensitivities when geometry changes are involved;
- incomplete sensitivities in which some derivatives are neglected; and
- how to evaluate Hessian-vector products.

Mesh Sensitivities

Motivation

Suppose the design parameters, α, control the shape of a geometry.

- Consequently, the domain of the boundary value problem will change during the optimization.
- The discretization, both R_h and J_h, must account for this change.

There are different ways of accounting for the change in the boundary that depend primarily on the underlying discretization.

- Move the mesh nodes to conform with the updated geometry.
- Regenerate the mesh.

We will consider the first now, and the second later.

Explicit Mesh Movement

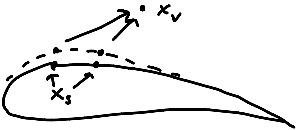
The simplest way to update the mesh when the boundary moves is to use an explicit mesh movement equation of the form

$$M(x,\alpha) = \begin{bmatrix} x_s - F_s(\alpha) \\ x_v - F_v(x_s) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- $x \in \mathbb{R}^m$ denote the mesh nodes
- $x_s \in \mathbb{R}^{m_s}$ are the surface mesh nodes, $x_s \in \partial\Omega$,
- $x_v \in \mathbb{R}^{m_v}$ are the volume mesh nodes, $x_v \in \Omega \setminus \partial\Omega$.

ie. $x_s = F_s(\alpha)$

$x_v = F_v(x_s)$



Explicit Mesh Movement (cont.)

We assume that the volume mesh nodes are explicit functions of the surface mesh nodes.

- Most mesh movement schemes have been developed for simulating a deforming surface in the absense of design parameters, e.g. for FSI, and only need the positions of x_s.

For explicit mesh movement, we can express the volume nodes in terms of α via

$$x_v - F_v(x_s) = x_v - F_v(F_s(\alpha)) = 0.$$

Explicit Mesh Movement (cont.)

So we can write all the nodes as explicit functions of α

$$M(x,\alpha) = x - F(\alpha) = 0.$$

- More generally (i.e. implicit mesh movement) we cannot write x_v in terms of α explicitly.

Explicit Mesh Movement (cont.)

Under explicit mesh movement, the total derivative can be written as

$$\frac{DJ_h}{D\alpha} = \frac{\partial}{\partial \alpha}\left(J_h(x,\alpha) + \psi_h^T R_h(x,\alpha)\right).$$

$$= \frac{\partial J_h}{\partial \alpha} + \frac{\partial J_h}{\partial x}\frac{\partial x}{\partial \alpha} + \psi_h^T \frac{\partial R_h}{\partial \alpha} + \psi_h^T \frac{\partial R_h}{\partial x}\frac{\partial x}{\partial \alpha}$$

$$= \frac{\partial J_h}{\partial \alpha} + \psi_h^T \frac{\partial R_h}{\partial \alpha} + \left[\frac{\partial J_h}{\partial x} + \psi_h^T \frac{\partial R_h}{\partial x}\right]\frac{\partial x}{\partial \alpha}$$

Explicit Mesh Movement (cont.)

Note the appeareance of vector-matrix products; these can be computed matrix-free using the reverse mode of AD

Example: Consider $\left[\psi_h^T \frac{\partial R_h}{\partial x}\right] \frac{\partial x}{\partial \alpha}$ compute left to right

1) Apply reverse mode to R_h
   - differentiate w.r.t. x, seed with ψ_h

2) Apply reverse mode to $F(\alpha) = x$
   - differentiate w.r.t. α, seed with $\psi_h^T \frac{\partial R_h}{\partial x}$ from step 1)

Implicit Mesh Movement

While explicit mesh movement algorithms are fast, most are not robust to large changes in shape.

Consequently, for more robust mesh movement, practictioners favor implicit mesh movement approaches like

- linear spring analogies;
- linear + torsional spring analogies; and
- linear/nonlinear elastic-solid analogies.

We will not review all the possiblities here, but will instead focus on sensitivity analysis in the context of implicit mesh movement.

Implicit Mesh Movement (cont.)

We will assume that the implicit mesh movement can be expressed as

$$M(x,\alpha) = \begin{bmatrix} x_s - F_s(\alpha) \\ F_v(x_v, x_s) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

- We continue to assume that the surface nodes are explicit functions of the parameters, which is true for all geometry parameterizations I am aware of.
- However, the volume nodes are coupled to one another, and are dependent on the surface nodes.

Implicit Mesh Movement (cont.)

An example of $F_v(x_v, x_s) = 0$ is given by the linear-elastic-solid analogy:

$$Lx_v = f(x_s),$$

- $L = L^T$ is the stiffness matrix that arises in (e.g.) the FEM discretization of the linear elasticity equations; and
- the load vector $f(x_s)$ is based on prescribing the displacements at the surface.

Implicit Mesh Movement (cont.)

For implicit mesh movement, we see that

$$\frac{\partial x_v}{\partial x_s} = -\left(\frac{\partial F_v}{\partial x_v}\right)^{-1} \frac{\partial F_v}{\partial x_s}.$$

surface nodes

- To get all the sensitivities requires m_s solves with $\partial F_v/\partial x_v$.
- Given the potentially large number of surface nodes, this is not practical.

comes from total derivative of $F_v = 0$:

$$\frac{d}{dx_s} F_v = \frac{\partial F_v}{\partial x_s} + \left(\frac{\partial F_v}{\partial x_v}\right)\frac{D x_v}{D x_s} = 0$$

Implicit Mesh Movement (cont.)

The solution is to treat the mesh-movement equation as an additional constraint and introduce mesh adjoints into the Lagrangian:

$$L_h = J_h + \psi_h^T R_h + \lambda^T F_v.$$

Then the first-order optimality conditions give us

$$\frac{\partial L_h}{\partial \psi_h} = R_h = 0 \quad ; \quad \frac{\partial L_h}{\partial \lambda} = F_v = 0 \qquad \text{(BVP and mesh move.)}$$

$$\frac{\partial L_h}{\partial u_h} = \frac{\partial J_h}{\partial u_h} + \psi_h^T \frac{\partial R_h}{\partial u_h} = 0 \qquad \text{(BVP adjoint eqn)}$$

$$\frac{\partial L_h}{\partial x_v} = \frac{\partial J_h}{\partial x_v} + \psi_h^T \frac{\partial R_h}{\partial x_v} + \lambda^T \frac{\partial F_v}{\partial x_v} = 0 \qquad \text{(mesh adjoint eqn)}$$

$$\frac{\partial L_h}{\partial \alpha} = \frac{\partial J_h}{\partial \alpha} + \psi_h^T \frac{\partial R_h}{\partial \alpha} + \lambda^T \frac{\partial F_v}{\partial \alpha} = \frac{D J_h}{D \alpha} \qquad \text{(total deriv.)}$$

Implicit Mesh Movement (cont.)

Thus, the mesh-adjoint equation is

$$\left(\frac{\partial F_v}{\partial x_v}\right)^T \lambda = -\frac{\partial J_h}{\partial x_v} - \psi_h^T \frac{\partial R_h}{\partial x_v}.$$

- Notice that we need the BVP adjoint in order to solve for the mesh adjoint.
- Makes sense; the adjoint process reverses the order of operations.

This idea, i.e. of introducing an adjoint, can be applied more generally for sensitivity analysis whenever there is an implicit dependence described by an equation.

Incomplete Sensitivities

Motivation

Computing mesh sensitivities can add expense and certainly complicates the process of evaluating total derivatives of functionals.

Notice, however, that the exact sensitivities based on the continuous BVP do not depend on a mesh. In other words

$$\frac{DJ_h}{D\alpha} = \frac{\partial J_h}{\partial \alpha} + \psi_h^T \frac{\partial R}{\partial \alpha} + \left[\frac{\partial J_h}{\partial x_v} + \psi_h^T \frac{\partial R}{\partial x_v} \right] \frac{\partial x_v}{\partial \alpha}$$
$$\rightarrow J'[\alpha] + (\psi, N'[\alpha](u, \alpha))_\Omega$$

$\rightarrow 0$ as $h \rightarrow 0$

- In the limit as $h \rightarrow 0$, the contribution due to the mesh sensitivities must go to zero.

Incomplete Mesh Sensitivities

Consequently, we could neglect the mesh sensitivities and the total derivative may still be sufficiently accurate (with respect to the infinite dimensional derivative).

Neglecting the mesh sensitivities arises in several context:

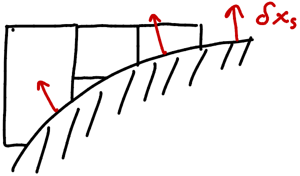
- Algorithms based on the continuous adjoint approach often (but not always) neglect the mesh sensitivities;
- When mesh regeneration is used during optimization, rather than mesh movement; and
- When cut-cell methods are used.

Incomplete Mesh Sensitivities (cont.)

Cut-cell methods are similar to mesh generation in the sense that a modified boundary can lead to a modified mesh topology.

- A change in mesh topology implies a discontinuity in the derivative, in general, which is problematic for most optimization algorithms.

Nevertheless, cut-cell researchers have had success in considering only infinitesimal changes when evaluating the mesh sensitivity, so that only cells/elements adjacent to the boundary are impacted.



assume shape change does not alter mesh topology (i.e. connectivity)

Incomplete Sensitivities, in General

The idea of neglecting certain terms in the total derivative can be applied more generally, and the resulting derivatives are called incomplete sensitivities or partial sensitivities.

- For example, one may "freeze" shock sensors or the turbulence eddy viscosity, thereby treating them as constant in some sense.

Such incomplete sensitivities were once common as a way of reducing coding effort and, sometimes, reducing cost.

- With the advent of AD and complex-step methods, the justification for incomplete sensitivities is reduced.

Example

The following results from [AB99] illustrate the potential risks of incomplete sensitivities.

- two-element airfoil at M=0.25, aoa=1 deg
- variables are the x and y position of the flap
- functional is the lift coefficient

variable	**with** $\partial/\partial x_v$	**without** $\partial/\partial x_v$
x translation	0.35481	8.4547
y translation	-7.1370	44.136

note magnitude and sign!

Hessian-vector Products

Motivation

Some optimization algorithms are based on Newton's method and need to solve linear systems of the form

$$Hp = -g,$$

- $g_i = DJ_h/D\alpha_i$ is the total derivative
- $H_{ij} = D^2 J_h / D\alpha_i D\alpha_j$ is the (total) Hessian
- p is the Newton step

For large-scale problems, the above equation is solved using iterative methods for which we need to evaluate Hessian-vector products Hz for some $z \in \mathbb{R}^n$.

Second-order Adjoints

We can use adjoints to compute Hessian-vector products.

Idea: define functional

$$\left(\frac{DJ_h}{D\alpha}\right)^T z,$$

then take the total derivative to get Hz.

- This is just like any other functional, such as J_h.
- However, we have to account for the fact that this functional depends on both u_h and the adjoint ψ_h.

$$\left(\frac{DJ_h}{D\alpha}\right)^T z = \left(\frac{\partial J_h}{\partial \alpha} + \psi_h^T \frac{\partial R_h}{\partial \alpha}\right)^T z$$

Second-order Adjoints (cont.)

Consequently, we have two constraints that need to be included in the Lagrangian:

$$R_h(u_h, \alpha) = 0$$

$$S_h(\psi_h, u_h, \alpha) \equiv \frac{\partial J_h}{\partial u_h} + \left(\frac{\partial R_h}{\partial u_h}\right)^T \psi_h = 0$$

Then we define

$$L(\alpha, u_h, \psi_h, w_h, \phi_h) = \underbrace{\left(\frac{DJ_h}{D\alpha}\right)^T z}_{\left(\partial J/\partial\alpha + \psi^T \partial R/\partial\alpha\right)^T z} + \phi_h^T R_h(u_h, \alpha) + w_h^T S_h(\psi_h, u_h, \alpha)$$

- $w_h \in \mathbb{R}^s$ and $\phi_h \in \mathbb{R}^s$ are sometimes called second-order adjoints.

Second-order Adjoints (cont.)

The desired Hessian-vector product is given by

$$\frac{\partial L}{\partial \alpha} = \left(\frac{\partial^2 J_h}{\partial \alpha^2} + \psi_h^T \frac{\partial^2 R_h}{\partial \alpha^2}\right)^T z + \phi_h^T \frac{\partial R_h}{\partial \alpha} + w_h^T \frac{\partial S_h}{\partial \alpha}$$

- If second-derivatives are not available, use directional finite-difference (no cost penalty).
- Still need w_h and ϕ_h.

Second-order Adjoints (cont.)

Equation for w_h:

$$\frac{\partial L}{\partial \psi_h} = \frac{\partial R_h}{\partial \alpha} z + w_h^T \frac{\partial R_h}{\partial u_h}^T = 0$$

$$\Rightarrow \quad \left(\frac{\partial R_h}{\partial u_h}\right) w_h = -\left(\frac{\partial R_h}{\partial \alpha}\right) z$$

This is a linearized state eqn.

- system matrix is BVP Jacobian

Second-order Adjoints (cont.)

Equation for ϕ_h:

$$\frac{\partial L}{\partial u_h} = \left(\frac{\partial^2 J}{\partial u_h \partial \alpha} + \psi_h^T \frac{\partial^2 R}{\partial u_h \partial \alpha} \right)^T z + \phi_h^T \frac{\partial R_h}{\partial u_h} + w_h^T \frac{\partial S_h}{\partial u_h} = 0$$

$$\Rightarrow \left(\frac{\partial R_h}{\partial u_h} \right)^T \phi_h = - \frac{\partial S_h}{\partial u_h}^T w_h - z^T \left(\frac{\partial^2 J}{\partial u_h \partial \alpha} + \psi_h \frac{\partial^2 R_h}{\partial u_h \partial \alpha} \right)$$

This is a reverse (ie. adjoint) problem

- system matrix is transposed BVP Jacobian
- second derivatives can again be evaluated with one directional FD.

References

[AB99] W. Kyle Anderson and Daryl L. Bonhaus, *Airfoil design on unstructured grids for turbulent flows*, AIAA Journal **37** (1999), no. 2, 185–191.