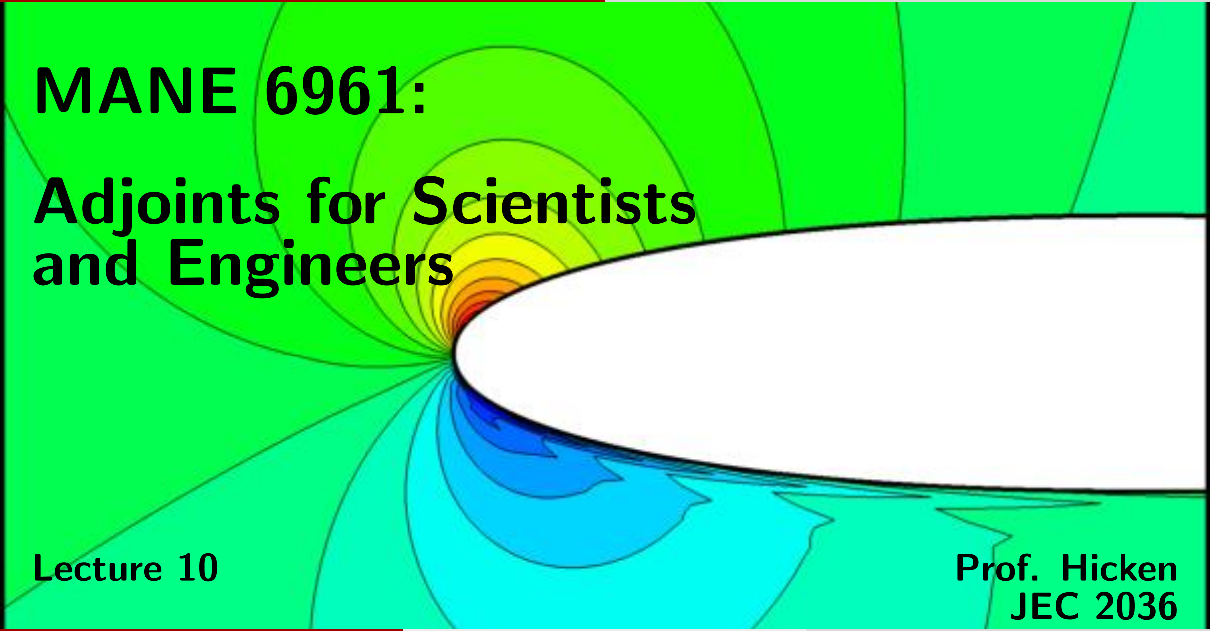

MANE 6961:

Adjoints for Scientists and Engineers

Lecture 10

Prof. Hicken
JEC 2036

Discrete Adjoint for Nonlinear Problems

Motivation

Recap: Being able to derive the adjoint BVP for nonlinear problems is useful because

- we can discretize the resulting adjoint BVP and use it to compute sensitivities and error estimates (i.e. the continuous adjoint approach); and
- the adjoint BVP provides insight into adjoint consistency, which is important for functional superconvergence and error estimation.

Nevertheless, deriving the adjoint BVP for complex nonlinear models — like turbulence models or constitutive relations — can be challenging and error prone.

Motivation (cont.)

It is for such complex nonlinear problems that the discrete adjoint approach is attractive, because it provides a straightforward, methodical approach.

- The discrete adjoint has other advantages we will discuss later in the context of sensitivity analysis.

In this lecture, we will review the discrete adjoint for nonlinear problems, and summarize different methods of computing the necessary Jacobian.

Derivation of the Discrete Adjoint

Let $R_h : \mathbb{R}^s \rightarrow \mathbb{R}^s$ be a nonlinear state equation, e.g. the discretization of a nonlinear BVP. We will assume that R_h is continuously differentiable.

Furthermore, we will assume that $u_h \in \mathbb{R}^s$ is the unique solution to the equation

$$R_h(u_h) = 0.$$

Finally, let $J_h : \mathbb{R}^s \rightarrow \mathbb{R}$ be a nonlinear, continuously differentiable function, e.g. a discretized functional.

Derivation of the Discrete Adjoint (cont.)

Recall that we derived the nonlinear adjoint BVP by differentiating the operators and applying Green's identity.

In the discrete case, Fréchet differentiation is equivalent to taking the partial derivatives:

$$R_h'[u_h] = \frac{\partial R_h}{\partial u_h},$$
$$J_h'[u_h] = \frac{\partial J_h}{\partial u_h}.$$

- $R_h'[u_h] = \partial R_h / \partial u_h$ is the Jacobian of the residual
- $J_h'[u_h] = \partial J_h / \partial u_h$ is the gradient of the function

Derivation of the Discrete Adjoint (cont.)

Furthermore, the analog of Green's identity is the bilinear identity,

$$(\psi_h, L_h u_h)_h = \left(L_h^T \psi_h, u_h\right)_h .$$

Substituting the Jacobian (and using v_h in place of u_h), we find

$$\left(\psi_h, \frac{\partial R_h}{\partial u_h} v_h\right)_h = \left(\frac{\partial R_h}{\partial u_h}^T \psi_h, v_h\right)_h .$$

Thus, we can make the following associations with the linear (discrete) adjoint problem from Lecture 4:

- $g_h \to \frac{\partial J_h}{\partial u_h}$; and
- $L_h^T \to \left(\frac{\partial R_h}{\partial u_h}\right)^T$.

Derivation of the Discrete Adjoint (cont.)

Definition: Discrete Adjoint Problem (nonlinear)

Let $u_h \in \mathbb{R}^s$ be the solution of $R_h(u_h) = 0$, where $R_h : \mathbb{R}^s \to \mathbb{R}^s$ is continuously differentiable. Then, the discrete adjoint problem associated with the continuously differentiable function $J_h : \mathbb{R}^s \to \mathbb{R}$ is

$$\left(\frac{\partial R_h}{\partial u_h}\right)^T \psi_h = \frac{\partial J_h}{\partial u_h}. \tag{DisAdj}$$

- As with the adjoint BVP, the discrete adjoint equation is a linear equation.

Evaluating the Jacobian for the Discrete Adjoint

Overview

In the context of BVPs, the discrete adjoint equation,

$$\left(\frac{\partial R_h}{\partial u_h}\right)^T \psi_h = \frac{\partial J_h}{\partial u_h}, \tag{DisAdj}$$

is usually a large, sparse linear system.

There are two challenges associated with this equation:

1. evaluating the transposed Jacobian (and gradient); and
2. solving the linear system.

For the remainder of this lecture, we will focus on the first of these. Next lecture we will discuss how to solve the system.

Jacobian Evaluation

The most onerous part of the discrete adjoint is forming the Jacobian, $\partial R_h / \partial u_h$, and, to a lesser extent, the gradient, $\partial J_h / \partial u_h$.

We will look at three ways of evaluating the Jacobian and/or gradient:

1. analytical evaluation;
2. brute-force evaluation; and
3. graph-based coloring evaluation.

Analytical Evaluation

For some BVP discretizations, it is straightforward to evaluate $\partial R_h / \partial u_h$ and $\partial J_h / \partial u_h$ analytically.

- That is, we "hand-code" the derivatives

Pros & Cons of analytical evaluation:

✓ usually produces the fastest code
✓ expert knowledge can take advantage of sparsity explicitly
✗ tedious, and expensive in terms of human effort
✗ error prone

Brute-force Evaluation

The simplest approach to finding the Jacobian and the gradient is a brute-force, column-by-column evaluation.

Let's consider how the complex-step method might be used to evaluate $\partial R_h / \partial u_h$ using such an approach. . .

Brute-force Evaluation (cont.)

```
dRdu = zeros(s, s) # storage for Jacobian
uh_c = zeros(Complex128, size(uh))
res_c = zeros(uh_c)
ceps = 1e-30
for i = 1:s # loop over state variables
  uh_c[i] += complex(0.0, ceps) # pert. state i
  calcResidual!(uh_c, res_c)    # eval. residual
  dRdu[:,i] = imag(res_c)./ceps # get column i
  uh_c[i] -= complex(0.0, ceps) # reset state i
end
```

- In the above code, the complex-step derivative approximation can be replaced with a finite-difference approximation, or the forward-mode of algorithmic differentiation.

Brute-force Evaluation (cont.)

Pros & Cons of brute-force evaluation:

- ✓ easy to implement
- ✓ can be applied to any residual
- ✗ costly: scales as s^2
- ✗ may not take advantage of sparsity

Coloring-based Evaluation

Consider the following nonlinear BVP:

$$\begin{aligned} u\frac{\partial u}{\partial x} &= f, \qquad \forall x \in [0,1], \\ u(0) &= u_L \end{aligned}$$

Suppose we discretize this problem using a finite-difference method as follows:

Coloring-based Evaluation (cont.)

With $s = 7$ state variables, it is easy to see that the Jacobian for the discretization has the following sparsity structure:

$$\frac{\partial R_h}{\partial u_h} = \begin{bmatrix} a_{11} & a_{12} & & & & & \\ a_{21} & a_{22} & a_{23} & & & & \\ & a_{32} & a_{33} & a_{34} & & & \\ & & a_{43} & a_{44} & a_{45} & & \\ & & & a_{54} & a_{55} & a_{56} & \\ & & & & a_{65} & a_{66} & a_{67} \\ & & & & & a_{76} & a_{77} \end{bmatrix}$$

Coloring-based Evaluation (cont.)

Now, in order to find the jth column of this Jacobian using the complex-step approximation, we perturb u_h by e_j, where e_j is the jth column of the $s \times s$ identity matrix:

$$\left(\frac{\partial R_h}{\partial u_h}\right)_{:,j} = \frac{\Im\left[R_h(u_h + e_j \epsilon i)\right]}{\epsilon},$$

where

$$(e_j)_k = \begin{cases} 1, & j = k, \\ 0, & j \neq k \end{cases}$$

Coloring-based Evaluation (cont.)

Suppose that, instead of perturbing in the direction e_j, we instead perturbed in the direction

$$d = \sum_{j \in I} e_j$$

where $I \subset \{1, 2, \ldots, s\}$ is a subset of unique state indices.

- Notice that d is a binary vector of 1s and 0s.

Perturbing by d gives us a sum of columns:

$$\frac{\Im\left[R_h(u_h + d\epsilon i)\right]}{\epsilon} = \sum_{j \in I} \left(\frac{\partial R_h}{\partial u_h}\right)_{:,j} = \left(\frac{\partial R_h}{\partial u_h}\right) d$$

Coloring-based Evaluation (cont.)

Idea: If we choose I (or, equivalently, d) carefully, then we can extract the entries of the Jacobian from the product $\frac{\partial R_h}{\partial u_h} d$.

Example:

$$\frac{\partial R_h}{\partial u_h} d = \begin{bmatrix} a_{11} & a_{12} & & & & & \\ a_{21} & a_{22} & a_{23} & & & & \\ & a_{32} & a_{33} & a_{34} & & & \\ & & a_{43} & a_{44} & a_{45} & & \\ & & & a_{54} & a_{55} & a_{56} & \\ & & & & a_{65} & a_{66} & a_{67} \\ & & & & & a_{76} & a_{77} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} \\ a_{21} \\ a_{34} \\ a_{44} \\ a_{54} \\ a_{67} \\ a_{77} \end{bmatrix}$$

Coloring-based Evaluation (cont.)

In the present example, we need only 3 directions in order to evaluate the Jacobian (independent of the size of s!).

$$\begin{aligned} d_1 &= \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \\ d_2 &= \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \\ d_3 &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}. \end{aligned}$$

In order for the above to work, each d_i must select *structurally orthogonal columns* from the Jacobian…

Coloring-based Evaluation (cont.)

Definition: Structually Orthogonal Columns

Consider a matrix $A \in \mathbb{R}^{n\times n}$. The columns $A_{:,i}$ and $A_{:,j}$ are structurally orthogonal if a non-zero entry in row k of one implies a zero entry in that same row for the other, i.e.

$$\begin{aligned} a_{ki} \neq 0 \quad &\Rightarrow \quad a_{kj} = 0, \\ a_{kj} \neq 0 \quad &\Rightarrow \quad a_{ki} = 0. \end{aligned}$$

This implies that $(A_{:,j})^T A_{:,i} = 0$.

Coloring-based Evaluation (cont.)

Using the concept of structurally orthogonal columns, the problem of finding the appropriate d_i to evaluate the Jacobian inexpensively can be stated succinctly as follows:

Given the sparsity structure of the Jacobian, find a structurally orthogonal partition of its columns that has the fewest groups.

- In the simple example, we identified a structurally orthogonal partition consisting of three groups.

Coloring-based Evaluation (cont.)

The above problem can be formulated as a graph-coloring problem.

- hence, existing graph-coloring libraries can be applied to find a structurally orthogonal partition of the columns of the Jacobian.

Pros & Cons of coloring-based evaluation:

✓ relatively easy to implement; can leverage existing coloring algorithms
✓ can be applied to any residual
✓ much faster than brute-force
✗ not as fast as "smart" analytical evaluations

Debugging Strategies

It is critically important to verify the Jacobian evaluation, especially when some form of analytical method is used.

Here I present a simple debugging strategy based on the bilinear identity. . .

Debugging Strategies (cont.)

Let $v_h \in \mathbb{R}^s$ and $z_h \in \mathbb{R}^s$ be arbitrary, given vectors.

- A good, default choice for v_h and z_h is to choose them to have Gaussian distributed entries.

Then we have the following (bilinear) identity:

$$z_h^T \left(\frac{\partial R_h}{\partial u_h} \right) v_h = v_h^T \left(\frac{\partial R_h}{\partial u_h} \right)^T z_h.$$

This simple identity is useful, because the left- and right-hand sides are evaluated using two independent methods.

Debugging Strategies (cont.)

On the left-hand side of the identity, we use a directional derivative to evaluate $\left(\frac{\partial R_h}{\partial u_h}\right) v_h$. For example

$$\left(\frac{\partial R_h}{\partial u_h}\right) v_h \approx \frac{\Im\left[R_h(u_h + v_h \epsilon i)\right]}{\epsilon},$$

or

$$\left(\frac{\partial R_h}{\partial u_h}\right) v_h \approx \frac{R_h(u_h + v_h \epsilon) - R_h(u_h)}{\epsilon}.$$

After we obtain the vector $\left(\frac{\partial R_h}{\partial u_h}\right) v_h$, we form the inner product with z_h.

Debugging Strategies (cont.)

On the right-hand side of the identity, we use the Jacobian that we evaluated using the analytical, brute-force, or coloring-based approach.

- I prefer to apply the transposed Jacobian to z_h first, provided the transpose operation has been independently verified.

If the two inner products do not match, you have a bug. With high probability, the bug is in the Jacobian evaluation.

- If the two products do match, that does not imply that the Jacobian is correct; however, the probability that it is correct is incredibly high given the random choice for v_h and z_h.

Debugging Strategies (cont.)

The above test provides a binary, “yes” or “no” answer to the question “is the Jacobian evaluation correct.”

However, the test can also be adapted to isolate the source of the problem if the answer turns out to be “no.”

- By setting $v_h = e_j$, we are able to select the jth column of the Jacobian.
- By setting $z_h = e_j$, we are able to select the jth row of the Jacobian.

This strategy can also be used to identify particular regions of the domain (e.g. boundaries) to see if the problem lies with the differentiation of particular terms.

Debugging Strategies (cont.)

Another debugging strategy can be used if the residual is composed of a sum of terms:

$$R_h(u_h) = \sum_{i=1}^{t} R_{h,i}(u_h).$$

- Such residuals often arise when there are different transport terms (advection and diffusion) and source terms in the PDE.
- If the code permits each term to be (independently) excluded from the residual, then the Jacobian associated with the individual terms can be verified separately to isolate bugs.

References