Electronic Transactions on Numerical Analysis.
Volume 32, pp. 134-144, 2008.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

ON THE EVALUATION OF FINITE ELEMENT SENSITIVITIES TO NODAL COORDINATES*

RENÉ SCHNEIDER† AND PETER K. JIMACK‡

Abstract. We present a derivation of the derivative of general systems of finite element equations with respect to the coordinates of the nodes in the underlying finite element mesh. The resulting expressions allow the systematic evaluation of such derivatives without the need to resort to automatic differentiation or the expense associated with finite difference approximations. The principal motivation for this work comes from problems in optimal design, however, other potential applications are also described. The results obtained are validated through numerical examples.

Key words. adjoint, discrete adjoint, sensitivity analysis, differentiation, derivatives, finite element method, FEM, moving finite elements, moving mesh, moving nodes, nodal coordinates, node movement, node positions, shape derivative

AMS subject classifications. 65N30, 80M10, 49Q10, 49Q12

1. Introduction.

This short paper is concerned with the evaluation of derivatives of finite element (FE) discretizations with respect to the positions of the nodes in the underlying FE mesh. This problem may arise in a number of different contexts, however, the primary motivation for our work comes from determining the sensitivity of a finite element solution, or some quantity derived from it, to the location of the mesh points. The need for such information arises naturally in the context of design optimization, where a domain shape is sought that yields the optimal performance of some functional of the solution to a partial differential equation (PDE), or system, defined on that domain, [12, 24]. Typical examples of this occur for fluid flow past a body, where one may wish to minimize the drag on the body or maximize the lift or down-force subject to given constraints, [9, 10, 11, 21]. As the shape of the domain is altered smoothly, the FE mesh covering the domain may also be smoothly perturbed and the sensitivity of the solution, or a derived quantity of interest, may be computed. If the solution is obtained via the finite element method, then it is clear, from the chain rule, that the derivative of various quantities with respect to the node positions are required.

Further details of this motivation are presented in Section 2 below. It should be noted, however, that the need to evaluate derivatives of finite element systems with respect to the node positions can arise in other contexts, too. For example, [15] illustrates how general moving grid methods for time-dependent equations and systems may exploit knowledge of derivatives of finite element basis functions with respect to node positions within their formulation. The literature is awash with such methods; see, e.g., [1, 16, 17, 20, 22]. However they are typically restricted to special cases such as linear elements on triangles, for which the differentiation is relatively straightforward [15]. Another potential application comes in research which seeks to establish locally optimal meshes for FE discretizations of elliptic variational problems; see, e.g., [3, 18, 26, 28]. Such problems may be posed by noting that the derivative of the corresponding energy functional with respect to each node position must be zero; see [3, 14].

Of course there are numerous ways in which derivatives of complex expressions with

*Received October 12, 2007. Accepted for publication May 5, 2008. Published online on March 4, 2009. Recommended by T. Apel.
†TU Chemnitz, Fakultät für Mathemetik, 09107 Chemnitz, Germany
(rene.schneider@mathematik.tu-chemnitz.de).
‡University of Leeds, School of Computing, Leeds LS2 9JT, UK (pkj@comp.leeds.ac.uk).

ETNA
Kent State University
etna@mcs.kent.edu

respect to variables such as node locations may be evaluated. Automatic differentiation (AD) has become a topic of much important research in recent years; see, e.g., [4, 8, 13, 27, 29] and many more. This approach allows large subprograms (typically functions) to be differentiated with respect to their input parameters, yielding new subprograms (for evaluating these derivatives) as their output. This technique is extremely powerful. In particular, for large legacy codes, it presents a favourable option; an overview of available AD software can be found on the internet[1].

However, there are a number of limitations. Even with AD the differentiation of large programs, such as PDE solvers, is not trivial. Defining the task for the AD software alone requires an inexperienced user to get familiar with the AD software, which may not be his primary interest. Further, the code produced by AD is not necessarily (or generally) as efficient as code that is hand-generated. Modern PDE solvers rely heavily on the utilisation of the problem structure to attain their efficiency. As [27, Section 4] describes, applying AD to a PDE solver in a black-box fashion is likely to produce suboptimal efficiency. It is only possible to get (near) optimal performance for the derivative code, if it uses the problem structure. AD has to be applied at just the right portions of the code, and/or the code should be tuned to give good performance with AD.

Thus, implementation of the discrete adjoint "by hand" remains an important option. Further, if approached in a systematic manner, exploiting the knowledge about the formulae that are implemented by the software, then it is a simpler task than one might think. Besides the of use of hand-calculated expressions for the derivatives, as described in this present work, these formulae may also be useful for theoretical analysis of the derivative information in the AD context.

An alternative to AD is simply to evaluate approximations to derivatives based upon finite differences, [7], or related schemes, [19]. This approach is of course easy to implement, simply requiring multiple calls to existing subprograms. However the computational expense when there are large numbers of independent variables can be prohibitive.

A very popular technique to overcome this problem associated with high dimensionality is the discrete adjoint approach [5, 6, 23]. This is introduced briefly in the following section for completeness. Using this technique the entire gradient of a performance functional may be evaluated at a cost comparable to just one additional evaluation of the functional itself. This is in contrast to finite difference approaches, which have a cost equivalent to $\mathcal{O}(m)$ functional evaluations, where m is the number of independent variables. In cases where we wish to compute a derivative with respect to the position of each node in a finite element mesh m is typically very large.

Having provided motivation for this work in Section 2, in Section 3 we present the main results of this paper, which provide a derivation of the derivatives of general finite element systems with respect to their node locations. Finally, in Section 4 we present a validation of the results and a discussion of their significance.

2. The Discrete Adjoint Method. Consider a finite element discretization of an elliptic PDE (say) on a domain whose geometry is uniquely defined by a set of parameters, $\mathcal{F}$. The FE solution is defined implicitly by the (possibly nonlinear) algebraic system

$$R(u(\mathcal{F}), \mathcal{F}) = 0, \tag{2.1}$$

where u is the vector of finite element coefficients and $R(\cdot, \cdot) = 0$ represents the FE equations on the domain defined by $\mathcal{F}$. Let $\widetilde{I}(u(\mathcal{F}), \mathcal{F})$ be a scalar-valued function, which depends

[1] See http://www.autodiff.org.

ETNA
Kent State University
etna@mcs.kent.edu

upon $u(\mathcal{F})$. Since $u(\mathcal{F})$ satisfies (2.1), we may regard this functional as depending only upon $\mathcal{F}$, that is:

$$I(\mathcal{F}) := \widetilde{I}(u(\mathcal{F}), \mathcal{F}), \tag{2.2}$$

so long as (2.1) holds.

Suppose now that we wish to investigate the sensitivity of the quantity I to perturbations in the parameters $\mathcal{F}$. We begin by considering the effect of small perturbations, $\delta\mathcal{F}$, of $\mathcal{F}$ in (2.2) and (2.1). Discarding higher order derivative terms, such a perturbation results in a perturbation δI in I,

$$\delta I = \frac{\partial \widetilde{I}}{\partial u}\delta u + \frac{\partial \widetilde{I}}{\partial \mathcal{F}}\delta\mathcal{F}, \tag{2.3}$$

where δu is defined by the perturbation of (2.1) which has to be zero, i.e.,

$$0 = \delta R = \frac{\partial R}{\partial u}\delta u + \frac{\partial R}{\partial \mathcal{F}}\delta\mathcal{F}. \tag{2.4}$$

As $\delta R = 0$, we can multiply it by an arbitrary term Ψ^T and subtract it from the right-hand side of (2.3), giving

$$\begin{aligned} \delta I &= \frac{\partial \widetilde{I}}{\partial u}\delta u + \frac{\partial \widetilde{I}}{\partial \mathcal{F}}\delta\mathcal{F} - \Psi^T\left(\frac{\partial R}{\partial u}\delta u + \frac{\partial R}{\partial \mathcal{F}}\delta\mathcal{F}\right) \\ &= \left(\frac{\partial \widetilde{I}}{\partial u} - \Psi^T\frac{\partial R}{\partial u}\right)\delta u + \left(\frac{\partial \widetilde{I}}{\partial \mathcal{F}} - \Psi^T\frac{\partial R}{\partial \mathcal{F}}\right)\delta\mathcal{F}. \end{aligned} \tag{2.5}$$

This implies that δI may be evaluated without calculating δu provided

$$\left[\frac{\partial R}{\partial u}\right]^T \Psi = \left[\frac{\partial \widetilde{I}}{\partial u}\right]^T, \tag{2.6}$$

and if $u(\mathcal{F})$ is well-defined by (2.1), then (2.6) uniquely defines Ψ, which is of the same dimension as u. (2.6) is known as the adjoint equation and Ψ as the adjoint solution. With this choice of Ψ the perturbation δI is

$$\delta I = \left(\frac{\partial \widetilde{I}}{\partial \mathcal{F}} - \Psi^T\frac{\partial R}{\partial \mathcal{F}}\right)\delta\mathcal{F},$$

revealing the representation of the total derivative

$$\frac{\mathrm{D}I}{\mathrm{D}\mathcal{F}} = \frac{\partial \widetilde{I}}{\partial \mathcal{F}} - \Psi^T\frac{\partial R}{\partial \mathcal{F}}. \tag{2.7}$$

The importance of this representation is that, once the original equation (2.1) is solved and $I(\mathcal{F})$ evaluated from (2.2), $\mathrm{D}I/\mathrm{D}\mathcal{F}$ may be evaluated for little more than the cost of a single solve of the linear system (2.6) and a single matrix-vector product in (2.7), regardless of the dimension of $\mathcal{F}$. This is to be compared to other methods for evaluating $\mathrm{D}I/\mathrm{D}\mathcal{F}$, which typically require the solution of (2.1) (or a linearised version) for each component of $\mathcal{F}$. Evaluating the derivative by means of (2.6) and (2.7) is called the discrete adjoint method; see [6] and the references therein for an introduction.

ETNA
Kent State University
etna@mcs.kent.edu

In the above derivation the specific definition of the parameter vector $\mathcal{F}$ is unimportant, and may be anything from coefficients of the PDE, to boundary data, or indeed the shape of the domain.

Now consider the case for which the dependence of the finite element solution upon $\mathcal{F}$ is via the FE mesh, as it is for example the case when $\mathcal{F}$ defines the shape of the domain. Suppose that the connectivity of this mesh remains constant but that as $\mathcal{F}$ is perturbed, the coordinates of the node points in the mesh are perturbed too, as a continuously differentiable function of $\mathcal{F}$. Let the vector s denote the nodal coordinates of each mesh vertex, then we may express $s = s(\mathcal{F})$. Furthermore

$$\frac{\partial R}{\partial \mathcal{F}} = \frac{\partial R}{\partial s} \cdot \frac{\partial s}{\partial \mathcal{F}} \qquad \text{and} \qquad \frac{\partial \widetilde{I}}{\partial \mathcal{F}} = \frac{\partial \widetilde{I}}{\partial s} \cdot \frac{\partial s}{\partial \mathcal{F}} \,. \tag{2.8}$$

Hence, assuming that the dependence of s upon $\mathcal{F}$ is known, or can be determined, the problem of evaluating $\mathrm{D}I/\mathrm{D}\mathcal{F}$ in (2.7) reduces to that of differentiating R and I with respect to the nodal locations s. This is the problem that is addressed in the following section.

One of the simplest alternative approaches to compute $\mathrm{D}I/\mathrm{D}\mathcal{F}$ that one may think of is to use finite differences, i.e., compute u and I for small perturbations of $\mathcal{F}$ and use the differences to approximate $\mathrm{D}I/\mathrm{D}\mathcal{F}$. This approach requires minimal work to program, but the number of times that (2.1) has to be solved is proportional to the dimension of $\mathcal{F}$. Thus if the number of parameters is large this approach is inefficient compared to the discrete adjoint method. The same applies to approaches which solve equations similar to the sensitivity equation (2.4) in order to compute $\partial u/\partial \mathcal{F}$ and then compute $\mathrm{D}I/\mathrm{D}\mathcal{F}$ by the chain rule. It should be noted however, that if one is interested in the derivatives of not only one dependent variable I but rather a vector of dependent variables, then the sensitivity equation approach is more efficient, if the number of dependent variables I is greater than the number of independent variables $\mathcal{F}$.

In order to see the link to AD, one should note that in AD one generally distinguishes two modes of operation, forward mode and reverse mode. Forward mode essentially applies the chain rule of differentiation to generate $\partial u/\partial \mathcal{F}$ as an intermediate result and therefore suffers from the same drawbacks as described above. However, reverse mode uses ideas similar to the discrete adjoint and thus (at least in theory) allows to evaluate $\mathrm{D}I/\mathrm{D}\mathcal{F}$ with a cost independent of the dimension of $\mathcal{F}$, with very little programming requirements. However, as already discussed in Section 1, some consideration on how to apply AD with optimal performance may still be necessary; see, e.g., [27].

3. Derivatives with respect to node positions. Equation (2.8) gives rise to the need to compute the Jacobian matrix $\partial R/\partial s$, i.e., the matrix of partial derivatives of the discrete residual vector $R(u, s)$ with respect to the node positions s is required. Let $R(u, s) = 0$ denote the finite element discretisation under consideration. The Jacobian matrix $\partial R/\partial s$ can be computed by summing up element contributions, in a similar way as the stiffness matrix is assembled.

REMARK 3.1. For the the discrete adjoint technique it is not necessary to compute and store the whole Jacobian, because this Jacobian is only used for a single matrix-vector product $\Psi^T(\partial R/\partial s)$ in (2.7). Thus, it is beneficial not to assemble the whole Jacobian, but to use the local contributions to compute the local matrix-vector products and assemble the resulting vector.

The canonical approach to computing $\partial R/\partial s$ is to consider the formulae by which the residual is computed, and to differentiate these. For finite element discretisations of second order elliptic PDE operators, the chain rule of differentiation can be used to reduce this to only two essential terms:

ETNA
Kent State University
etna@mcs.kent.edu

- the derivative of the gradients of the basis functions with respect to the node positions, and
- the derivative of the determinant of the element mapping with respect to the node positions.

In the following this is demonstrated in detail for the Poisson equation as model problem and explicit formulae are given for these two terms. Extension to other PDEs, for example the Lamé equation of linear elasticity or the Navier-Stokes equations, is straight forward.

REMARK 3.2. Of course it is also possible to use finite differences or automatic differentiation to compute $\partial R/\partial s$. However, these should be applied on a local level only in order to avoid an increase in computational complexity; see, e.g., [27].

REMARK 3.3. The derivative of the performance criterion $\partial\widetilde{I}/\partial s$ can be computed in the same way, if the performance functional is also an integral over a function of the solution and its spatial derivatives, which is usually the case.

Consider the Poisson equation with constant forcing term $f \equiv 1$. This gives rise to the discrete formulation

$$0 = R(u,s) = \left[a(u_h,\varphi_i) - b(\varphi_i)\right]_{i=1}^{N}, \tag{3.1a}$$

$$a(u,v) := \int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}\Omega, \tag{3.1b}$$

$$b(v) := \int_\Omega 1\, v \,\mathrm{d}\Omega, \tag{3.1c}$$

where $\left\{\varphi_i \in H_0^1(\Omega) : i = 1,\ldots,N\right\}$ form a basis of the finite element function space. The integrals contained in the definition of $a(.,.)$ and $b(.)$ are usually computed using numerical integration on each element and then summing over the elements. Hence, the i-th row of the discrete residual vector R may be expressed as

$$[R]_i = \sum_{\substack{T\in\mathcal{T}_h\\ s_i\in\overline{T}}} \sum_{j=1}^{n} \left(\sum_{\ell=1}^{m} w_\ell \left(\nabla_{\!T}\varphi^{(g_T(j))} \cdot \nabla_{\!T}\varphi^{(i)}\; u_{g_T(j)} - 1\,\widehat{\varphi}^{(i)}(\hat{x}_\ell)\right) |\det(J_T)| \right), \tag{3.2}$$

where the outermost sum is over those elements $T \in \mathcal{T}_h$ which have node i as a vertex, n is the number of Lagrange basis functions per element, $\hat{x}_\ell$ ($\ell = 1,\ldots,m$) are the quadrature points on the master element, and w_ℓ are the quadrature weights. Further, $g_T(j)$ is the global node number of vertex j of element T, $\nabla_{\!T}$ denotes the gradient restricted to element T evaluated at the point x_ℓ (which corresponds to $\hat{x}_\ell$ via the subparametric or isoparametric element mapping), $\widehat{\varphi}^{(j)}(\hat{x})$ is the basis function corresponding to node j evaluated at point $\hat{x}$ within the master element, and J_T is the Jacobian of the element mapping. In this expression the parts that are non-constant with respect to the node positions are

$$\nabla_{\!T}\varphi^{(g_T(j))}, \quad \nabla_{\!T}\varphi^{(i)}, \quad \text{and } |\det(J_T)|,$$

in each element T. Applying the product rule of differentiation gives for the derivative of

$[R]_i$ with respect to $[s_k]_t$, the t-th coordinate of node k,

$$
\begin{aligned}
\frac{\partial [R]_i}{\partial [s_k]_t} = \sum_{\substack{T\in\mathcal{T}_h:\\ (s_i\in\overline{T}\\ \wedge s_k\in\overline{T})}} \sum_{j=1}^{n} \Bigg[\sum_{\ell=1}^{m} w_\ell \Big(\frac{\partial \nabla_T \varphi^{(g_T(j))}}{\partial [s_k]_t} \cdot \nabla_T \varphi^{(i)} \\
+ \nabla_T \varphi^{(g_T(j))} \cdot \frac{\partial \nabla_T \varphi^{(i)}}{\partial [s_k]_t} \Big) |\det(J_T)| u_{g_T(j)} \\
+ \sum_{\ell=1}^{m} w_\ell \big(\nabla_T \varphi^{(g_T(j))} \cdot \nabla_T \varphi^{(i)} u_{g_T(j)} \\
- 1\, \widehat{\varphi}^{(i)}(\hat{x}_\ell) \big) \frac{\partial |\det(J_T)|}{\partial [s_k]_t} \Bigg].
\end{aligned}
\tag{3.3}
$$

Assuming that each of the individual terms in (3.3) can be evaluated, it is now a straightforward task to compute $\partial R/\partial s$ (or $\Psi^T \partial R/\partial s$).

It remains to derive expressions for the derivatives of the gradient $\nabla_T \varphi^{(i)}$ and the determinant $|\det(J_T)|$. The following propostion presents these results.

PROPOSITION 3.4.

$$
\frac{\partial [\nabla_T \varphi^{(i)}]_u}{\partial [s_k]_t} = -\left[\nabla_T \psi^{(k)}\right]_u \left[\nabla_T \varphi^{(i)}\right]_t, \tag{3.4}
$$

$$
\frac{\partial |\det(J_T)|}{\partial s_{g_T(k)}} = |\det(J_T)| J_T^{-T} \hat{\nabla} \widehat{\psi}^{(k)}(\hat{x}_\ell). \tag{3.5}
$$

Proof. Let us recall some properties of subparametric[2] finite elements (see, e.g., [2]): the definition of the element mapping function $M_T(.)$ (3.6) by means of the shape functions on the master element $\widehat{\psi}$, the resulting expression for the Jacobian J_T of the element mapping (3.7), the definition of the ansatz functions $\varphi^{(i)}$ on the world element by means of the ansatz functions $\widehat{\varphi}^{(i)}$ on the reference element (3.8), and the resulting expression for the world element gradient $\nabla_T \varphi^{(i)}(x)$ (3.9):

$$
x = M_T(\hat{x}) := \sum_{k=1}^{n} s_k \widehat{\psi}^{(k)}(\hat{x}), \tag{3.6}
$$

$$
J_T = \left[\frac{\partial x}{\partial \hat{x}}\right] = \sum_{k=1}^{n} s_k \left[\hat{\nabla} \widehat{\psi}^{(k)}(\hat{x})\right]^T, \tag{3.7}
$$

$$
\varphi^{(i)}(x) := \widehat{\varphi}^{(i)}(M_T^{-1}(x)), \tag{3.8}
$$

$$
\nabla_T \varphi^{(i)}(x) = J_T^{-T} \hat{\nabla} \widehat{\varphi}^{(i)}(M_T^{-1}(x)). \tag{3.9}
$$

By analogy with (3.8) and (3.9) we define

$$
\psi^{(i)}(x) := \widehat{\psi}^{(i)}(M_T^{-1}(x)), \tag{3.10}
$$

$$
\nabla_T \psi^{(i)}(x) = J_T^{-T} \hat{\nabla} \widehat{\psi}^{(i)}(M_T^{-1}(x)). \tag{3.11}
$$

[2]The clasical Lagrange finite elements belong to this class of elements, for isoparametric elements one simply subsitutes $\psi = \varphi$, $\widehat{\psi} = \widehat{\varphi}$.

Here s_k denotes the coordinate vector of the k-th node of the element T and $\hat{\nabla}$ denotes the gradient on the reference element, i.e., the partial derivatives with respect to the reference element coordinates $\hat{x}$. Two immediate consequences are

$$\begin{aligned} \frac{\partial \left[J_T^T\right]_{(v,j)}}{\partial [s_k]_t} &= \sum_{r=1}^{n} \delta_{k,r}\delta_{t,j} \left[\hat{\nabla}\widehat{\psi}^{(r)}(\hat{x})\right]_v && \text{(by (3.7))} \\ &= \delta_{t,j} \left[\hat{\nabla}\widehat{\psi}^{(k)}(\hat{x})\right]_v , \end{aligned} \tag{3.12}$$

where $\delta_{i,j}$ denotes the Kronecker delta.

Due to the application of the numerical integration based on the reference element, the world gradient is only evaluated for points corresponding to the quadrature points on the reference element $\hat{x}_\ell$,

$$\nabla_{\!T}\varphi^{(i)} := \nabla_{\!T}\varphi^{(i)}(M_T(\hat{x}_\ell)) = J_T^{-T}\hat{\nabla}\widehat{\varphi}^{(i)}(\hat{x}_\ell). \tag{3.13}$$

Since $\hat{\nabla}\widehat{\varphi}^{(i)}(\hat{x}_\ell)$ is independent of the node positions, $\nabla_{\!T}\varphi^{(i)}$ depends on the node positions only via the transpose of the element Jacobian J_T^T. The derivatives with respect to the node positions can be calculated using the implicit function theorem on the reformulated (3.13),

$$0 = J_T^T([s_k]_t)\nabla_{\!T}\varphi^{(i)} - \hat{\nabla}\widehat{\varphi}^{(i)}(\hat{x}_\ell). \tag{3.14}$$

This gives

$$\frac{\partial \nabla_{\!T}\varphi^{(i)}}{\partial [s_k]_t} = -J_T^{-T}\frac{\partial J_T^T}{\partial [s_k]_t}\nabla_{\!T}\varphi^{(i)}, \tag{3.15}$$

where the derivative of the transposed Jacobian is in the component wise sense. Taking a closer look at the derivative of the u-th component of $\nabla_{\!T}\varphi^{(i)}$, we get

$$\begin{aligned} \frac{\partial [\nabla_{\!T}\varphi^{(i)}]_u}{\partial [s_k]_t} &= -\sum_{v=1}^{d}\left[J_T^{-T}\right]_{(u,v)}\sum_{j=1}^{d}\frac{\partial \left[J_T^T\right]_{(v,j)}}{\partial [s_k]_t}\left[\nabla_{\!T}\varphi^{(i)}\right]_j \\ &= -\sum_{v,j=1}^{d}\left[J_T^{-T}\right]_{(u,v)}\frac{\partial \left[J_T^T\right]_{(v,j)}}{\partial [s_k]_t}\left[\nabla_{\!T}\varphi^{(i)}\right]_j \\ &= -\sum_{v,j=1}^{d}\left[J_T^{-T}\right]_{(u,v)}\delta_{t,j}\left[\hat{\nabla}\widehat{\psi}^{(k)}(\hat{x}_\ell)\right]_v\left[\nabla_{\!T}\varphi^{(i)}\right]_j && \text{(by (3.12))} \\ &= -\sum_{v=1}^{d}\left[J_T^{-T}\right]_{(u,v)}\left[\hat{\nabla}\widehat{\psi}^{(k)}(\hat{x}_\ell)\right]_v\left[\nabla_{\!T}\varphi^{(i)}\right]_t \\ &= -\left[\nabla_{\!T}\psi^{(k)}\right]_u\left[\nabla_{\!T}\varphi^{(i)}\right]_t . && \text{(by (3.11))} \end{aligned}$$

This shows the first part of the proposition.

The derivative of $|\det(J_T)|$ also can be reduced to a derivative of J_T due to the identity

$$\left[\frac{\partial}{\partial a_{i,j}}\det(A)\right] = A^{-T}\det(A),$$

ETNA
Kent State University
etna@mcs.kent.edu

which follows from expansion to sub-determinants and the adjoint representation of the inverse of the matrix A. Applying this formula and an analogue of (3.12), we get

$$\begin{aligned}\frac{\partial}{\partial[s_k]_t}|\det(J_T)| &= |\det(J_T)| \sum_{u,v=1}^{d} \left[J_T^{-T}\right]_{u,v} \frac{\partial[J_T]_{u,v}}{\partial[s_k]_t} \\ &= |\det(J_T)| \sum_{u,v=1}^{d} \left[J_T^{-T}\right]_{u,v} \delta_{t,u} \left[\hat{\nabla}\widehat{\psi}^{(k)}(\hat{x}_\ell)\right]_v \\ &= |\det(J_T)| \sum_{v=1}^{d} \left[J_T^{-T}\right]_{t,v} \left[\hat{\nabla}\widehat{\psi}^{(k)}(\hat{x}_\ell)\right]_v . \qquad \square\end{aligned}$$

4. Numerical verification. In [25, Section 2.7.2.2], for verification purposes, derivative values obtained by the discrete adjoint method, using the expressions derived in Section 3, have been compared to those obtained by the finite difference approximation

$$\frac{\mathrm{D}I}{\mathrm{D}\mathcal{F}_i} \approx \frac{I(\mathcal{F}_i + h) - I(\mathcal{F}_i - h)}{2\,h} \tag{4.1}$$

(CD, central difference), with $h = 10^{-5}$. The examples used in [25] use Taylor-Hood (P_2/P_1) finite element discretisations of the incompressible Navier-Stokes equations. Here we give results obtained for Example 2.2 from [25], because these are representative for many more tests which have been undertaken, and because they highlight the advantage of the discrete adjoint method compared to finite differences.

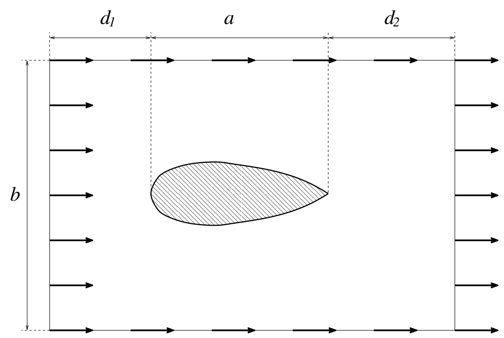

FIGURE 4.1. *Example: Obstacle in a channel.*

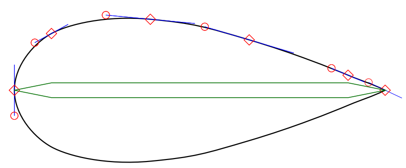

FIGURE 4.2. *Parameterised obstacle shape (NACA0040).*

The example comprises of an obstacle moving in a channel of viscous fluid at steady state at Reynolds number $Re = 20$. Under consideration are the derivatives of the drag with respect to shape parameters defined by Bezier splines, resulting in ten parameters. See Figure 4.1 for the domain geometry, and Figure 4.2 for an illustration of the shape parameterisation.

ETNA
Kent State University
etna@mcs.kent.edu

As the number of shape parameters is very moderate in comparison to the number of mesh points in this example, we follow the suggestion of an anonymous referee to include a further alternative, where in the adjoint representation (2.7) the derivatives $\partial I/\partial\mathcal{F}$ and $\Psi^T(\partial R/\partial\mathcal{F})$ are evaluated by central differences. This allows to better judge the cost of computing those derivatives by the methodology from Section 3 in comparison to the overhead of the mesh deformation. Values computed by this approach will be denoted by CDR.

TABLE 4.1
Verification of adjoint derivative evaluation in the FE discretisation by comparison to finite difference values.

adjoint	CD	CDR	(CD-adj)/|CD|
`1.799e-01`	`1.799e-01`	`1.799e-01`	`3.9e-05`
`4.912e-01`	`4.912e-01`	`4.912e-01`	`1.3e-05`
`1.118e+00`	`1.118e+00`	`1.118e+00`	`3.8e-06`
`7.675e-01`	`7.675e-01`	`7.675e-01`	`2.7e-06`
`-6.314e-02`	`-6.311e-02`	`-6.314e-02`	`-4.9e-04`
`-3.683e-02`	`-3.677e-02`	`-3.683e-02`	`-1.4e-03`
`-5.934e-02`	`-5.934e-02`	`-5.934e-02`	`-4.6e-05`
`-1.702e-01`	`-1.702e-01`	`-1.702e-01`	`-2.5e-05`
`2.665e-01`	`2.665e-01`	`2.665e-01`	`2.9e-05`
`1.644e-02`	`1.644e-02`	`1.644e-02`	`-1.8e-05`
$Re = 20$ $(\nu = 0.05)$, $\#DOFs = 572608$, $t_{\text{solve}} =$ `3.9e+03s`			
$t_{\text{adj}} =$ `2.6e+03s`	$t_{\text{CD}} =$ `2.3e+04s`	$t_{\text{CDR}} =$ `3.6e+03s`	

Table 4.1 lists the derivative values as computed by the discrete adjoint method, central difference (CD), the approach described above (CDR), and the relative difference between the adjoint, and central difference values. The tests have been performed on a Linux desktop PC with Intel(R) Core(TM)2 CPU 6400 at 2.13GHz, using the single threaded C code `FEINS` of the first author, using meshes producing the number of degrees of freedom as listed in the table ($\#DOFs$). The derivative values compare well and the most significant relative errors occur only for those components of the gradient with small absolute value. In those cases the relative error of the finite difference approximation is largest due to cancellation effects. Note that smaller values of h ($\ll 10^{-5}$) in (4.1) led to unreliable results for the finite difference approximation due to cancellation effects and the inexact solves of the discretised Navier-Stokes systems. The differences between the adjoint and CDR values are significantly smaller, as they use the same (inexact) solution of the adjoint equation (2.6) and avoid the differentiation of the inexact solution of the discrete Navier-Stokes system.

Table 4.1 also shows the total time taken for the computation of the value of I (t_{solve}) and those taken for a subsequent derivative computation by each method (t_{adj}, t_{CD}, and t_{CDR}, respectively). Note the different exponent in t_{CD}. The computation by the central differences already takes advantage of the small perturbations in the nonlinear system, initialising the perturbed nonlinear solves with the unperturbed solution, thus resulting in $t_{\text{CD}} \approx 5.9 \cdot t_{\text{solve}}$ rather than the $20 \cdot t_{\text{solve}}$ one might expect. As the discrete adjoint technique requires only one solution of the linear adjoint equation (2.6), rather than multiple solutions of linearised equations in Newton's method for solving the nonlinear Navier-Stokes system, it can be observed that $t_{\text{adj}} \approx 0.67 \cdot t_{\text{solve}}$, even less than t_{solve}. Overall, the advantage of the adjoint method is evident.

REMARK 4.1. A comparison with automatic differentiation is omitted here, but may be a subject of future work.

For more details on the example problem and its implementation, as well as further examples, see [25].

5. Conclusions. A systematic approach to differentiation of finite element discretisations with respect to the coordinates of the nodes in the FE mesh has been presented. The problem can be reduced to the derivatives of only two terms depending on the mesh geometry, for which expressions have been presented in Section 3. Results obtained with the presented approach have been verified by comparison to those obtained with finite difference approximation for an example problem. The example also demonstrates the efficiency of the methods discussed in this work.

Applications of the result range from shape optimisation to moving mesh FE discretisations, and possibly many more.

REFERENCES

[1] M. Baines, M. Hubbard, and P. Jimack, *A moving mesh finite element algorithm for the adaptive solution of time-dependent partial differential equations with moving boundaries*, Appl. Numer. Math., 54 (2005), pp. 450–469.
[2] P. Ciarlet, *The Finite Element Method for Elliptic Problems*, Classics in Applied Mathematics, SIAM, Philadelphia, 2002.
[3] M. Delfour, G. Payre, and J. Zolesio, *An optimal triangulation for 2nd-order elliptic problems*, Comput. Methods Appl. Mech. Engrg., 50 (1985), pp. 231–261.
[4] M. Fagan and A. Carle, *Reducing reverse-mode memory requirements by using profile-driven checkpointing*, Future Generation Computer Systems, 21 (2005), pp. 1380–1390.
[5] M. Giles, M. Duta, J.-D. Muller, and N. Pierce, *Algorithm developments for discrete adjoint methods*, AIAA J., 41 (2003), pp. 198–205.
[6] M. Giles and N. Pierce, *An introduction to the adjoint approach to design*, Flow Turbul. Combust., 65 (2000), pp. 393–415.
[7] P. Gill, W. Murray, and M. Wright, *Practical Optimization*, Academic Press, San Diego, 1981.
[8] A. Griewank, *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation*, SIAM, Philadelphia, 2000.
[9] M. Gunzburger, *Perspectives in Flow Control and Optimization*, SIAM, Philadelphia, 2003.
[10] M. Gunzburger and S. Manservisi, *Flow matching by shape design for the Navier-Stokes system*, in Optimal Control of Complex Structures, K. H. Hoffmann, et. al., eds., Birkhäuser, Basel, 2000, pp. 279–289.
[11] J. Hämäläinen, R. Mäkinen, and P. Tarvainen, *Optimal design of paper machine headboxes*, Internat. J. Numer. Methods Fluids., 34 (2000), pp. 685–700.
[12] J. Haslinger and R. Mäkinen, *Introduction to Shape Optimization*, SIAM, Philadelphia, 2003.
[13] P. Hovland, B. Norris, and B. Smith, *Making automatic differentiation truly automatic: coupling PETSc with ADIC*, Future Generation Computer Systems, 21 (2005), pp. 1426–1438.
[14] P. Jimack, *An optimal finite element mesh for elastostatic structural analysis problems*, Comput. & Structures, 64 (1997), pp. 197–208.
[15] P. Jimack and A. Wathen, *Temporal derivatives in the finite-element method on continuously deforming grids*, SIAM J. Numer. Anal., 28 (1991), pp. 990–1003.
[16] D. Lynch and K. O'Neill, *Continuously deforming finite elements for the solution of parabolic problems with and without phase change*, Internat. J. Numer. Methods Engrg., 17 (1981), pp. 81–96.
[17] A. Madzvamuse, P. Maini, and A. Wathen, *A moving grid finite element method for the simulation of pattern generation by turing models on growing domains*, J. Sci. Comput., 24 (2005), pp. 247–262.
[18] R. Mahmood and P. Jimack, *Locally optimal unstructured finite element meshes in 3 dimensions*, Comput. & Structures, 82 (2004), pp. 2105–2116.
[19] J. Martins, P. Sturdza, and J. Alonso, *The connection between the complex-step derivative approximation and algorithmic differentiation*, AIAA J., 0921 (2001).
[20] K. Miller and R. Miller, *Moving finite elements, I*, SIAM J. Numer. Anal., 18 (1981), pp. 1019–1032.
[21] B. Mohammadi and O. Pironneau, *Applied Shape Optimization for Fluids*, Oxford Science Publications, Oxford, 2001.
[22] M. Mosher, *A variable node finite element method*, J. Comput. Phys., 57 (1985), pp. 157–187.
[23] A. Noack and A. Walther, *Adjoint concepts for the optimal control of burgers equation*, Comput. Optim. Appl., 36 (2007), pp. 109–133.
[24] O. Pironneau, *Optimal Shape Design for Elliptic Systems*, Springer-Verlag, Berlin, 1984.

ETNA
Kent State University
etna@mcs.kent.edu

[25] R. SCHNEIDER, *Applications of the Discrete Adjoint Method in Computational Fluid Dynamics*, PhD thesis, University of Leeds, 2006. Available at http://www.engineering.leeds.ac.uk/comp/research/pubs/theses.shtml

[26] R. SCHNEIDER AND P. JIMACK, *Toward anisotropic mesh adaption based upon sensitivity of a posteriori estimates*, School of Computing Research Report Series 2005.03, University of Leeds, 2005. Available at http://www.comp.leeds.ac.uk/research/pubs/reports/2005/2005_03.pdf

[27] E. TIJSKENS, D. ROOSE, H. RAMON, AND J. D. BAERDEMAEKER, *Automatic differentiation for solving nonlinear partial differential equations: an efficient operator overloading approach*, Numer. Algorithms, 30 (2002), pp. 259–301.

[28] Y. TOURIGNY AND F. HULSEMANN, *A new moving mesh algorithm for the finite element solution of variational problems*, SIAM J. Numer. Anal., 35 (1998), pp. 1416–1438.

[29] A. WALTHER, *Automatic differentiation of explicit Runge-Kutta methods for optimal control*, Comput. Optim. Appl., 36 (2007), pp. 83–108.