Short paper proceedings of "Third ISSMO/ UBCAD/ UB/ AIAA World Congress on Structural and Multidisciplinary Optimization", Amherst, NY, May, 1999 (CD-ROM, 5 pp.)

Combining Automatic Derivatives and Hand-coded Derivatives in Sensitivity Analysis for Shape Optimization Problems

Raino A. E. Mäkinen

Faculty of Information Technology, University of Jyväskylä, Finland

1. Abstract

A hybrid approach for shape design sensitivity analysis for a class of shape optimization is presented. Hand-coded derivatives and automatic derivatives are combined in such a way that the adjoint equation technique can be utilized. This approach yields significant reduction in the memory and time required to compute derivatives if compared to the approach where automatic differentiation is applied to the whole code. Numerical examples are given.

2. Keywords

Sensitivity analysis, Shape Optimization, Automatic Differentiation

3. Problem Formulation

We consider the following abstract shape optimization problem in discrete form

$$\min_{\boldsymbol{a}\in\mathbb{R}^N} J_0(\boldsymbol{a}) = \mathcal{I}_0(\boldsymbol{a},\boldsymbol{q}(\boldsymbol{a})) \tag{1}$$

$$\text{subject to} \tag{2}$$

$$J_i(\boldsymbol{a}) = \mathcal{I}_i(\boldsymbol{a},\boldsymbol{q}(\boldsymbol{a})) \le 0,\ i = 1,...,m \tag{3}$$

$$\boldsymbol{R}(\boldsymbol{a},\boldsymbol{q}) = 0. \tag{4}$$

Here $\boldsymbol{a}$ is the vector of design parameters defining the shape $\Omega_{\boldsymbol{a}}$ of the system. The state $\boldsymbol{q}(\boldsymbol{a})$ of the system is obtained by solving the state equation (4).

Although gradient free global optimization methods like Genetic Algorithms are becoming more and more popular also in shape optimization, they cannot compete efficient gradient based methods like Sequential Quadratic Programming when good initial guess is known and high accuracy is requested. Unfortunately, gradient based methods require the computation of the partial derivatives of cost and constraint functions with respect to design variables. Finite difference approximations for derivatives may not be accurate and are obtained too slowly. Therefore, obtaining analytic derivatives is an important step in the numerical optimization process.

By shape design sensitivity analysis we mean computing derivatives of the functionals J_i with respect to geometrical parameters defining the positions of nodal coordinates of the finite element mesh. Hand-coding of shape design sensitivity analysis has been considered extremely elaborate and difficult even for linear state problems. This is probably due to the bad form in which most of the sensitivity formulae are presented. In these formulae there are usually too much explicit dependence on certain application or element type. This implies nonstructured programs which are difficult to debug and maintain. Despite this, hand-coded sensitivity analysis can be done with a reasonable amount of work even in the case of quite complicated state problems [1], [2], [3]. However, fully hand-coded derivatives are not always feasible at all, e.g. if unstructured and/or adaptive mesh generator is used.

4. On automatic differentiation of programs

Automatic differentiation (AD) is a technique for augmenting computer programs with derivative computations [4]. It exploits the fact that every computer program executes a sequence of elementary arithmetic operations. By applying the chain rule of derivative calculus repeatedly to these operations, accurate derivatives of arbitrary order can be computed automatically. Automatic differentiation has two basic modes the "forward" and "reverse" modes. The running time and storage requirements of the forward mode is approximately proportional to the number of design variables. The reverse mode which is closely related to adjoint methods has a lower operations count for derivative computations, but potentially very large memory requirements.

Automatic differentiation can be implemented in two different ways: Existing analysis code written in e.g. Fortran 77 is precompiled using a precompiler (like ADIFOR [5]) into a new code that includes derivative calculations, or operator overloading technique available in e.g. Fortran 90 is used to produce sensitivity information. Modern programming languages like Fortran 90 make it possible to redefine the meaning of elementary operators. That is, we can define a new type for floating point numbers that has gradient information associated with it. For each elementary operation and standard function (`+`, `*`, `sin()`, `dot_product()`,...) we can define the meaning of the operation for variables of that new data type. The advantages of operator overloading is that it almost completely hides the AD tool from the user. If the implementation of the AD tool is changed the source code needs no modification.

Although AD techniques can be applied to add gradient computations to codes in their entirety, significant reduction in the memory and time required to compute derivatives are of the possible if hand-coding is coupled with AD techniques. Therefore we proceed with the following hybrid approach. First, we implement the forward mode of AD using operator overloading in Fortran 90. Secondly, we develop the geometric sensitivity analysis with respect to positions of nodal coordinates in general matrix form for a class of elliptic partial differential equations.

5. Hybrid approach to sensitivity analysis

Quasilinear state problem

Many physical phenomena in elasticity, fluid dynamics and electromagnetics can be modeled using the second order quasilinear partial differential equation of type

$$\nabla \cdot \sigma + g = 0, \qquad \sigma = \rho(u)\nabla u \tag{5}$$

with suitable boundary conditions. The discrete analogue of (5) reads

$$\boldsymbol{R}(\boldsymbol{a};\boldsymbol{q}) \equiv \boldsymbol{K}(\boldsymbol{a};\boldsymbol{q})\,\boldsymbol{q} - \boldsymbol{f}(\boldsymbol{a}) = 0, \tag{6}$$

where $\boldsymbol{K}$ and $\boldsymbol{f}$ are the "stiffness" matrix and "force" vector, respectively.

Differentiating (6) implicitly gives

$$\frac{\partial \boldsymbol{R}(\boldsymbol{a};\boldsymbol{q})}{\partial a_k} + \frac{\partial \boldsymbol{R}(\boldsymbol{a};\boldsymbol{q})}{\partial \boldsymbol{q}}\frac{\partial \boldsymbol{q}}{\partial a_k} = 0. \tag{7}$$

By introducing an adjoint state vector $\boldsymbol{p}$ we can eliminate the partial derivative of $\boldsymbol{q}$ from (7) and obtain

$$\left(\frac{\partial \boldsymbol{R}(\boldsymbol{a};\boldsymbol{q})}{\partial \boldsymbol{q}}\right)^{\mathrm{T}} \boldsymbol{p} = \nabla_{\boldsymbol{q}}\mathcal{I}(\boldsymbol{a},\boldsymbol{q}) \tag{8}$$

$$\frac{\partial J(\boldsymbol{a})}{\partial a_k} = \frac{\partial \mathcal{I}(\boldsymbol{a},\boldsymbol{q})}{\partial a_k} - \boldsymbol{p}^{\mathrm{T}}\left(\frac{\partial \boldsymbol{R}}{\partial a_k}\right) \tag{9}$$

In existing FE-software hand-coded Jacobian $\partial\boldsymbol{R}/\partial\boldsymbol{q}$ is often available as it is required for the Newton–Raphson method. Also hand-coding the righthand side of (8) is usually not more difficult than coding the cost function $\mathcal{I}(\boldsymbol{a},\boldsymbol{q})$ itself.

The derivatives in (9) are more difficult due to the dependencies

$$\boldsymbol{a} \to \boldsymbol{X}(\boldsymbol{a}) \to \boldsymbol{R}(\cdot, \boldsymbol{q}),\ \mathcal{I}(\cdot, \boldsymbol{q}) \tag{10}$$

which imply that one must first differentiate the nodal coordinate matrix $\boldsymbol{X}(\boldsymbol{a})$ with respect to design parameters. Hand-coding of this is error-prone and sometimes even impossible. Therefore, in our hybrid approach the derivatives on the right hand side of (9) are computed using AD.

Below is a Fortran 90-style pseudocode for calculating the value of the cost function and its gradient at a given point `aa`. We assume that three-noded elements are used and there is one degree of freedom per node. The Fortran 90-module `AD` contains the code needed to implement a new data type `DVAR` and overload the arithmetic operations and standard functions for this new data type. Also mixed `DVAR`/`REAL` arithmetic is implemented. Moreover, it contains the functions `AD_indep_var`, `AD_gradient` and `AD_pd` which declare independent variable vector, return the gradient or partial derivative of a variable of type `DVAR`, respectively.

```
subroutine objfun( n, aa, I, dI )
USE AD
real, intent(IN) :: aa(n)
real, intent(OUT):: I, dI(:)

real:: Pe(3)
TYPE(DVAR):: X(Nnodes,2), Re(3), II
real, dimension(Nnodes,Nnodes):: Jac
real, dimension(Nnodes):: Q, dQ, P, R, dIQ
a = AD_indep_var( aa )
Call Generate_Mesh( a, X )
Q = 0.0
do while ( .not.converged )
   Call Calc_Jacobian_and_Residual( X, Q, Jac, R )
   Call LinearSolve( Jac, dQ, -R )
   Q = Q + dQ
end do
Call CostFunction( X, Q, II, dIQ )
Call LinearSolve( Transpose(Jac), P, dIQ )
I  = ad_value( II )
dI = ad_gradient( II )
do e=1,Nelems
   Pe = gather( P, e )
   call Calc_Element_Residual( e, X, Q, Re )
   do k=1,N
      dI(k) = dI(k) - dot_product( Pe, AD_pd( Re, k ) )
   end do
end do
end subroutine Objfun
```

<u>Linear state problem</u>
If the state problem (5) is linear then the sensitivity of the residual can be hand-coded in a straight-forward and portable way. Let us assume for simplicity that ρ and g are constants. Then using n-noded isoparametric elements the stiffness matrix and force vector corresponding the element Ω_e are given by

$$\boldsymbol{K} = \int_{\hat{\Omega}} \rho \boldsymbol{B}^{\mathrm{T}} \boldsymbol{B} |\boldsymbol{J}| \, dx_h, \qquad \boldsymbol{f} = \int_{\hat{\Omega}} g \boldsymbol{N} |\boldsymbol{J}| \, dx_h, \tag{11}$$

where $\hat{\Omega}$ is the reference element and $|\boldsymbol{J}|$ is the Jacobian of the mapping $\hat{\Omega} \to \Omega_e$.

For the sensitivity of the "strain–displacement" matrix $\boldsymbol{B}$ and the Jacobian determinant we have [6], [7]

$$\frac{\partial \boldsymbol{B}}{\partial a_k} = -\boldsymbol{B}\frac{\partial \boldsymbol{X}^e}{\partial a_k}\boldsymbol{B}, \qquad \frac{\partial |\boldsymbol{J}|}{\partial a_k} = |\boldsymbol{J}| \sum_{j=1}^{n}\sum_{i=1}^{2} \frac{\partial \varphi_j}{\partial x_i}\frac{\partial X_{ji}}{\partial a_k} \tag{12}$$

and

$$\frac{\partial \boldsymbol{K}^e}{\partial a_k} = \int_{\hat{\Omega}} \left[\rho \left(\frac{\partial \boldsymbol{B}}{\partial a_k} \right)^{\mathrm{T}} \boldsymbol{B}\,|\boldsymbol{J}| + \rho\,\boldsymbol{B}^{\mathrm{T}} \frac{\partial \boldsymbol{B}}{\partial a_k} |\boldsymbol{J}| + \rho\,\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B} \frac{\partial |\boldsymbol{J}|}{\partial a_k} \right] dx_h \tag{13}$$

$$\frac{\partial \boldsymbol{f}^e}{\partial a_k} = \int_{\hat{\Omega}} g\boldsymbol{N} \frac{\partial |\boldsymbol{J}|}{\partial a_k}\, dx_h \tag{14}$$

Matrix $\partial \boldsymbol{X}^e / \partial a_k$ is produced by the AD augmented mesh generator. The remaining terms in formulae (12)–(14) are not more difficult to hand-code than the original finite element code.

6. Numerical examples

Optimal design of a magnet

A classical problem in magnet design is that of finding the pole profile of, for example, a nonlinear H-shaped magned, in order to have a desired uniform field in the shaded subregion of the air gap [8]. The pole profile (shown in Figure 1) is given by a Bezier-curve defined by four design variables. Comparison of AD computed gradients and finite difference (FD) approximations with steplength 10^{-5} are shown in Table 1. Here $\partial_k AD$ and $\partial_k FD$ stand for the value of $\frac{\partial J}{\partial a_k}$ obtained using AD and forward difference approximation for $\frac{\partial J}{\partial a_k}$, respectively.

Although the mesh is unstructured reliable derivatives were obtained.

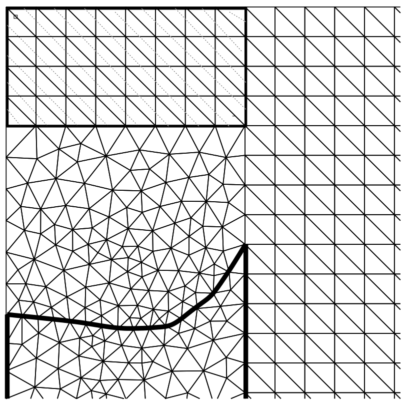
Figure 1.
Moving part of the FE-mesh

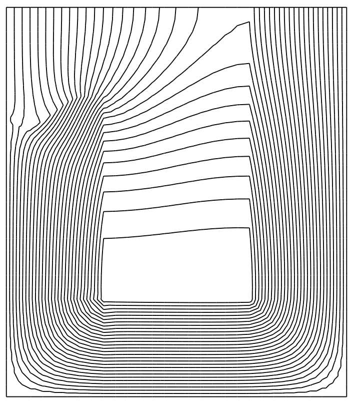
Figure 2.
Scalar magnetic potential contours

k	$\partial_k AD / \partial_k FD$
1	1.0000026
2	1.0000039
3	1.0000021
4	0.99999884

Table 1.

Optimal design of a hollow shaft

The design objective is to choose the outer shape of a hollow shaft that has a given cross-sectional area and has a maximal torsional stiffness. Due to symmetry, only quarter of the cross-section is considered. The outer boundary is parametrized using Bezier curve defined by five design variables. Comparison of AD computed gradients and finite difference (FD) approximations with steplength 10^{-4} are shown in Table 2.

k	$\partial_k AD/\partial_k FD$
1	1.0000523
2	0.99999747
3	0.99997835
4	0.99997738
5	0.99999647

Table 2.

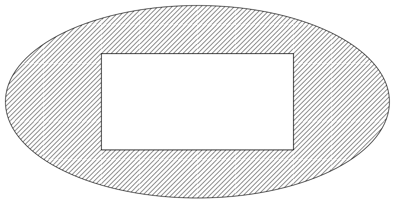
Figure 3.
Cross-section of the shaft.

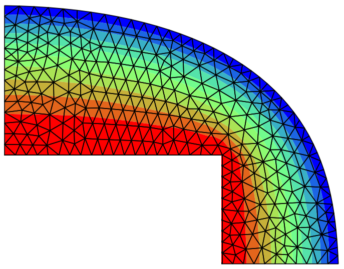
Figure 4.
FE-mesh and stress potential contours.

7. Conclusions

The hybrid method for shape design sensitivity analysis is both easy to program and efficient in terms of computer time and memory. It is efficient as the differentiation of the (non)linear state solver is avoided making is possible to use standard software (LAPACK, for example) to solve the linearized state problem. Computed sensitivities are very accurate provided that the mesh is fine enough and the nonlinear state equation is solved with sufficiently strict stopping criterion. Our approach is general as it applies to multidisciplinary shape optimization problems and several finite elements. General purpose programs can be easily developed as the dependence on the specific application can be isolated into separate modules.

8. References

[1] J. Haslinger and R. A. E. Mäkinen (1992), Shape optimization of elasto-plastic bodies under plane strains: sensitivity analysis and numerical implementation, *Structural Optimization*, **4**, 133–141.

[2] R. A. E. Mäkinen and J. Toivanen (1994), Optimal shape design for Helmholtz/potential flow problem using fictitious domain method, *Proceedings of 5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 529–536.

[3] P. Tarvainen, R. A. E. Mäkinen and J. Hämäläinen (1998), Shape optimization for laminar and turbulent flows with applications to geometry design of paper machine headboxes, *Proceedings of the Tenth International Conference on Finite Elements in Fluids*, M. Hafez and J. C. Heinrich (eds.), The University of Arizona, 536–541.

[4] A. Griewank (1989), On Automatic Differentiation, in *Mathematical Programming: Recent Developments and Applications*, M. Iri and K. Tanabe (eds.), Kluwer Academic Publishers, Amsterdam, 83–108.

[5] C. Bischof, A. Carle, P. Khademi, and A. Mauer (1996), ADIFOR 2.0: Automatic Differentiation of Fortran 77 Programs, *IEEE Computational Science & Engineering*, **3**, 18–32.

[6] R. A. Brockman (1987), Geometric sensitivity analysis with isoparametric finite elements, *Commun. Appl. Numer. Methods*, **3**, 495–499.

[7] R. Mäkinen (1990), Finite element design sensitivity analysis for nonlinear potential problems, *Commun. Appl. Numer. Methods*, **6**, 343–350.

[8] O. Pironneau (1984), Optimal Shape Design for Elliptic Systems, Springer-Verlag.