INTRODUCTION TO AND IMPLEMENTATION OF A DISCRETE ADJOINT CFD CODE

Charlie J. Anderson

Department of Aerospace Engineering, University of Bristol, Queen's Building, University Walk, Bristol, BS8 1TR, UK

ABSTRACT

*A technical **ABSTRACT** of maximum 200 words should be written centred, in 11pt Times New Roman italics. A concise and factual abstract is required, outlining in a single paragraph the aims, scope and conclusions of the paper. In a minute or less a reader can learn the rationale behind the study, general approach to the problem, pertinent results, and important conclusions or new questions. Remember, an abstract is often presented separately from the article, so it must be able to stand alone. For this reason, References should be avoided, but if essential, then cite the author(s) and year(s). Also, non-standard or uncommon abbreviations should be avoided, but if essential they must be defined at their first mention in the abstract itself. Leave one 11pt space after the abstract.*

Keywords: Adjoint method, Automatic differentiation, CFD, Aerodynamic Optimisation

1 INTRODUCTION

11pt space after this title. The purpose of an introduction is to acquaint the reader with the rationale behind the work, with the intention of defending it. It places your work in a theoretical context, and enables the reader to understand and appreciate your objectives.

The function of the Introduction is to:

- Establish the context of the work being reported. This is accomplished by discussing the relevant primary research literature (with citations) and summarizing our current understanding of the problem you are investigating;
- State the purpose of the work in the form of the hypothesis, question, or problem you investigated; and,
- Briefly explain your rationale and approach and, whenever possible, the possible outcomes your study can reveal.
- By the end of the first page, the reader should understand what the paper is trying to do.

Quite literally, the Introduction and/or Literature Review must answer the questions, "What was I studying? Why was it an important question? What did we know about it before I did this study? How will this study advance our knowledge?"

1.1 Project overview

Calculating the sensitivity, or gradient, of an aerodynamic cost function is of vital importance in optimisation problems. While the common method of evaluating it using finite differences is straightforward it is very expensive. Large optimisation problems can have 10^3 to 10^4 design variables, meaning the cost of $N_{\text{design vars}} \times \texttt{COST}(\text{F})$ for a single run can be prohibitive in efforts to run gradient-based optimisations.

The adjoint method offers an alternative to finite differences when calculating the sensitivity. While adjoint code is difficult to develop, it is far cheaper to gain the sensitivity. The adjoint method uses the chain rule to de-couple the large number of design variables from the expensive flow solver. Instead these partial derivatives are calculated separately, then the information is re-used in a more efficient manner. A full description of the method is set out in Section XX.

A literature review will be conducted after the introduction of the mathematics.

Most journal papers are aimed at academics with years of experience in the field and sacrifice some explanatory material for the sake of being concise. However this paper will be aimed at an individual who is skilled as computing and engineering but has little experience with the adjoint method.

2 THE ADJOINT EQUATIONS

2.1 A note on derivatives

Consider a function $f : \mathbb{R}^2 \to \mathbb{R}$ which is the function of two purely independent variables t and s. To find the variation with respect to t, the partial derivative is used.

$$f(t, s) = ts + t^2 \tag{1}$$

$$\begin{aligned} \frac{\partial f}{\partial t} &= \frac{\partial f}{\partial t} \\ &= s + 2t \end{aligned} \tag{2}$$

Now consider the same function $f : \mathbb{R}^2 \to \mathbb{R}$. While this also has two *independent* variables t and s, this has three inputs as it includes $x = t^2$. As it is now a composite function, the total derivative must be used.

$$f(t, s, x(t)) = ts + x \tag{3}$$

$$\begin{aligned} \frac{df}{dt} &= \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x}\frac{\partial x}{\partial t} \\ &= s + (1) \cdot (2t) \end{aligned} \tag{4}$$

Both equations evaluate to the same result provided the correct derivative is used. Notice that the $\frac{\partial f}{\partial t}$ term on the RHS of each equation actually means something different in both instances. Care must be taken to use the correct derivative in the correct situation.

One can ask why certain parts of t are abstracted into a new variable x. In an expensive cost calculation some parts are more expensive than others. In CFD calculating the flow variables is more expensive than other operations involving the design variables. The adjoint method works by separating the expensive variables from the rest of the calculation and making a substitution in order to allow information to be reused.

In the normal process of using finite differences the cost function is not often stated in terms of the flow variables because they are created then discarded in the process of finding the gradient without ever being explicitly stated outside of the CFD calculation. However here they are exposed as part of the derivation. It may appear if an extra term has appeared, but really it is the first term split into two, with the first partial derivative not equal from equation to equation.

$$\underset{\text{wrt } \alpha_i}{\texttt{SENS}} \left(J_r\right) = \text{do that bracket thing}$$

2.2 Linear alegbra setup

Often in the field of optimisation it is necessary to evaluate terms in the form of $g^T u$ which is the product of two vectors. The vector u can be seen as the solution of a linear system of equations A and vector f.

$$Au = f$$

There also exists the dual form for adjoint solution v and original vector g:

$$A^T v = g$$

Then by the adjoint equivalence we can show that

$$g^T u = v^T f$$

By making a substitution $v^T f = v^T(Au) = (A^T v)^T u = g^T u$. This equivalence forms the basis of the adjoint method. When evaluating the sensitivity of cost function J with respect to desgn variables α there is a term in the form of $g^T u$. With a knowledge of the dual form there are multiple ways of approaching this problem.

$$\begin{aligned} \frac{dJ}{d\alpha} &= \frac{\partial J}{\partial \alpha} + \frac{\partial J}{\partial w}\frac{\partial w}{\partial \alpha} \\ &= \frac{\partial J}{\partial \alpha} + g^T u \end{aligned} \tag{5}$$

$$\begin{aligned} \frac{dR}{d\alpha} &= \frac{\partial R}{\partial \alpha} + \frac{\partial R}{\partial w}\frac{\partial w}{\partial \alpha} = 0 \\ &= f - Au \end{aligned} \tag{6}$$

Now the classic optimisation problem has been rewritten in terms of the abstracted variables decisions can be made to improve the efficiency of the sensitivity evaluation. To get u, the governing equations can be solved as they are in a process known as the direct method.

$$\text{The direct method:} \quad \textit{solve } Au = f \quad \textit{where } \tfrac{\partial R}{\partial w} u = -\tfrac{\partial R}{\partial \alpha}$$

This is normally best used when the number of dependent variables N_J are much larger than the number of independent design variables N_α. However in the case of most optimisation problems this is not the case - the is often only one cost function and up to thousands of design variables. In this case it is better to use the adjoint method:

$$\text{The adjoint method:} \quad \textit{solve } A^T v = g \quad \textit{where } \tfrac{\partial R}{\partial w}^T v = -\tfrac{\partial J}{\partial w}^T$$

The sensitivity of the cost function can now be found by using $v^T f$ in place of $g^T u$ by the adjoint equivalence proved earlier.

2.3 The Adjoint Equations

Consider a CFD code developed to calculate body forces on an aerofoil (or any aerodynamic problem more generally). After solving the residual R for the flow vector w, the cost function J can be calculated, whether this is C_D, Breguet range, or something else. Both of these are functions of the design variables α.

$$\begin{aligned} \text{const function} \quad & \mathrm{J}(\alpha,\mathrm{w}) \qquad & J : \mathbb{R}^{N_\alpha+N_w} \to \mathbb{R}^{N_J} \\ \text{subject to} \quad & \mathrm{R}(\alpha,\mathrm{w}) = 0 \quad & R : \mathbb{R}^{N_\alpha+N_w} \to \mathbb{R}^{N_w} \end{aligned}$$

It is often important to calculate the sensitivity of the cost function, whether for gradient-based optimisation or something else. As ony design variables are input into the code it is found that $J = J(\alpha)$. The sensitivity is thus an array of partial derivatives.

$$\texttt{SENS}\big(J_r\big) = \frac{\partial J_r}{\partial \alpha_i}$$

$$\texttt{COST}\big(\texttt{SENS}\big) = N_\alpha \cdot \texttt{COST}\big(J_{\text{full}}\big) + 1$$

This results in a very costly operation, as a full CFD analysis must be run for each design variable perturbation for a first-order result. The adjoint method works by de-coupling the expensive calculation from the large number of design variables.

This time let the cost function be $J = J(\alpha, w(\alpha))$. This is exactly the same equation as before, with only the design variables α being input by the user, but an abstraction is being made in order to extract the flow vector. As discussed before, the vector of sensitivities evaluates to exactly the same thing, but it is split into a form convenient for the calculations,

$$\texttt{SENS}\big(J_r\big) = \frac{dJ_r}{d\alpha_i} = \frac{\partial J_r}{\partial \alpha_i} + \frac{\partial J_r}{\partial w_j}\frac{\partial w_j}{\partial \alpha_i}$$

A substitution is performed in order to decouple the design variables from the full CFD calculation. This is done by calculating the total derivative of the governing equations.

$$\frac{dR_s}{d\alpha_i} = \frac{\partial R_s}{\partial \alpha_i} + \frac{\partial R_s}{\partial w_j}\frac{\partial w_j}{\partial \alpha_i} = 0$$

As the governing equations are equivalent to zero, they can be added onto the main set of equations with a Lagrange multiplier λ_{rs}. This can be set to any value, so a convenient one is chosen to remove dependence on flowfield variables w.

$$\frac{dJ_r}{d\alpha_i} = \frac{\partial J_r}{\partial \alpha_i} - \lambda_{rs}\frac{\partial R_s}{\partial \alpha_i}$$

$$\lambda_{rs} = \frac{\partial J_r}{\partial w_j}\left[\frac{\partial R_s}{\partial w_j}\right]^{-1}$$

$$\texttt{COST}\big(\texttt{SENS}\big) = N_\alpha \cdot \texttt{COST}\big(J_{\text{full}}\big) + N_w \cdot \texttt{COST}\big(w\big)$$

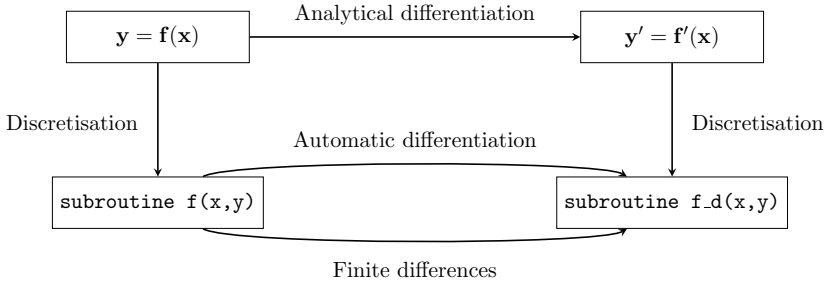
2.4 Discrete and continuous adjoint equations

3 AUTOMATIC DIFFERENTIATION

3.1 Introduction to AD

Automatic differentiation (also known as AD or algorithmic differentiation) is a process used to calculate the Jacobian of a given computer program. It is most effective when applied to a completely self-contained program with no external "black-box" functions as it will be a series of arithmetic operations $+, -, *, /$ and basic functions $\exp(), \sin(), \log()$.

There are two ways of doing AD. The most common method by far is source code transformation. This takes in the source code and returns new source code that produces the Jacobian of the specified variables. While development of the AD engine itself is difficult, it is very easy for the user to use. The other method of AD is operator overloading which is done by defining a new number type that contains both the number and it's derivative. As all of the work is done by the compiler operator overolading can result in much simpler implementation, however it is currentlly much slower than source code transformation, especially for reverse mode.



In the process of creating CFD codes the equations are derived then discretised, a process shown on the left hand side of Figure. For gradient based methods this needs to be taken one step further and have a corresponding sensitivity vector evaluated. There are several different ways of doing this. Analytical differentiation is possible but difficult for a large code and prone to errors. For this reason the method of finite differences is often used as it is very easy to do. There are many downside however, as there is a truncation error involved and it is very expensive. AD provides the best of both worlds as it is an exact solution and easy to do. The real advantage of AD however is the computational cost - if it is used correctly (in the forward or reverse mode) the cost can be almost independent of the independent or dependent variables respectively.

3.2 Front End - How to use it

When explaining AD many texts focus on how the chain rule is traversed to describe the properties. However the author feels that this approach is unhelpful as it explains neither the speed difference between forward and reverse mode, or how a novice would use an AD tool. For this reason, a "front end" description will be given for knowledge of how to use AD and what the effects are and a "back end" description to explain how it works.

Consider a function $F(x) : \mathbb{R}^n \rightarrow \mathbb{R}^m$ with n inputs and m outputs. The goal of AD is to calculate the system Jacobian (or sensitivities). If done by analytical differentiation or finite differences, this would cost , which is very expensive if there are many variables in the function.

$$J_{ij} = \frac{\partial F_i}{\partial x_j} = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \frac{\partial F_m}{\partial x_2} & \cdots & \frac{\partial F_m}{\partial x_n} \end{bmatrix}$$

$$\texttt{COST}(J) = n \cdot m \cdot \texttt{COST}(F)$$

The most intuitive mode of AD is the forward mode, also known as the tangent mode. This works by calculating all of the derivatives with respect to a sing fixed input, meaning that the cost of a single sweep is *independent of the number of output variables* m. For a situation where $m < n$ this is preferable. There is a slight scaling of this shown by variable C. Normally this is about 2 with a proven upper bound of 3.

As can be seen from equation , a forward mode sweep is equivalent to calculating one column of the Jacobian at a time. Which column is determined by a seed vector $\dot{x} = x_{seeds}$. The non-zero entry of this determines which input variable sensitivities $\dot{F} = F_{sens}$ are with respect to. This can be represented by $\dot{F}_i = \frac{\partial F_i}{\partial x_j} \cdot \dot{x}_j$, or equation.

$$\begin{bmatrix} \dot{F}_1 \\ \dot{F}_2 \\ \dot{F}_3 \end{bmatrix} = \begin{bmatrix} ? & \frac{\partial F_1}{\partial x_2} & ? & ? \\ ? & \frac{\partial F_2}{\partial x_2} & ? & ? \\ ? & \frac{\partial F_3}{\partial x_2} & ? & ? \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\texttt{COST}(J) = n \cdot C \cdot \texttt{COST}(F)$$

Reverse mode, also known as the adjoint mode calculates all of the derivatives with respect to a fixed output, or a row of the Jacobian. The adjoint variables $\bar{x} = x_{sens}$ are once again determined by the seeds $\bar{F} = F_{seeds}$ and can be represented by $\bar{x}_j = \frac{\partial F_i}{\partial x_j} \cdot \bar{F}_i$, equation.

$$\begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_3 \\ \bar{x}_4 \end{bmatrix} = \begin{bmatrix} ? & ? & ? & ? \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \frac{\partial F_2}{\partial x_3} & \frac{\partial F_2}{\partial x_4} \\ ? & ? & ? & ? \end{bmatrix}^T \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\texttt{COST}(J) = m \cdot C \cdot \texttt{COST}(F)$$

As there only needs to be m sweeps to calculate the Jacobian, it can be seen that the cost is*independent of the number of input variables* n. As a result, reverse mode is of much interest in the optimisation community as there is often only one output cost function but up to thousands of input design variables. The cost of one reverse sweep is higher than one forward - C is closer to 4 or 5, which means for a case where $m \approx n$ then forward mode is more efficient to use.

3.3 Back End - How the sweeps work

3.3.1 Set-up

During most of the AD examples the same test function is used. This can also be represented by a function tree.

$$f(x, y) = xy + sin(x)$$

ref Griewank 2000

ref Tapenade page?

Write the equivalent of both this stuff tbh.

3.3.2 Forward Mode

3.3.3 Reverse Mode

4 LITERATURE REVIEW

4.1 Literature review - Early

A comprehensive overview of adjoint methods for design pre-2000 has been written by M. Giles and N. Pierce [1].

This field has it's origins in optimal control from the work of Lions [2].

The first application to fluid mechanics was performed by Pironneau [3].

4.2 Literature review - Intermediate

1996 Jameson et al, ASO of compex aircraft configurations using the adjoint method [4]

1998 Giles, On the properties of solutions of the adjoint Euler equations [5]

1999 Jameson et al, Multipoint ASO using adjoint formulation and parallel computers [6]

2003 Jameson, Aerodynamic shape optimisation using the adjoint method [7]

2003 Martins, The complex-step derivative approximation [8]

2008 Martins, ADjoint: An approach for the rapid development of discrete adjoint solvers [9]

2011 JD Mueller, Verification of AD derived CFD codes [10]

4.3 Literature review - Cutting edge

Conditional statements for AD

Turbulence models for adjoint

5 IMPLEMENTATION

5.1 The CFD code

5.2 The Adjoint code

1. Create CFD code
2. Find partial derivatives $\frac{\partial J}{\partial \alpha}$, $\frac{\partial J}{\partial w}$, $\frac{\partial R}{\partial \alpha}$ and $\frac{\partial R}{\partial w}$
3. Solve adjoint equation for vector λ
4. Form total derivative to get the sensitivity

6 RESULTS

7 CONCLUSION

REFERENCES

[1] M. B. Giles and N. A. Pierce, "An introduction to the adjoint approach to design," *Flow, turbulence and combustion*, vol. 65, no. 3-4, pp. 393–415, 2000.

[2] J. L. Lions, *Optimal Control of Systems Governed by Partial Differential Equations (Grundlehren der Mathematischen Wissenschaften)*, vol. 170. Springer Berlin, 1971.

[3] O. Pironneau, "On optimum design in fluid mechanics," *Journal of Fluid Mechanics*, vol. 64, no. 1, pp. 97–110, 1974.

[4] J. Reuther, A. Jameson, J. Farmer, L. Martinelli, and D. Saunders, "Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation," in *34th Aerospace Sciences Meeting and Exhibit*, p. 94, 1996.

[5] M. B. Giles and N. A. Pierce, "On the properties of solutions of the adjoint euler equations," *Numerical Methods for Fluid Dynamics VI. ICFD*, pp. 1–16, 1998.

[6] J. J. Reuther, A. Jameson, J. J. Alonso, M. J. Rimlinger, and D. Saunders, "Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 1," *Journal of aircraft*, vol. 36, no. 1, pp. 51–60, 1999.

[7] A. Jameson, "Aerodynamic shape optimization using the adjoint method," *Lectures at the Von Karman Institute, Brussels*, 2003.

[8] J. R. Martins, P. Sturdza, and J. J. Alonso, "The complex-step derivative approximation," *ACM Transactions on Mathematical Software (TOMS)*, vol. 29, no. 3, pp. 245–262, 2003.

[9] C. A. Mader, J. RA Martins, J. J. Alonso, and E. V. Der Weide, "Adjoint: An approach for the rapid development of discrete adjoint solvers," *AIAA journal*, vol. 46, no. 4, pp. 863–873, 2008.

[10] F. Christakopoulos, D. Jones, and J.-D. Müller, "Pseudo-timestepping and verification for automatic differentiation derived cfd codes," *Computers & Fluids*, vol. 46, no. 1, pp. 174–179, 2011.